To

Hervé (1921-2004) *and* *Pierre* (1922-2004)

The Importance of Embodiment

Towards Co-operation in

Multi Robot Systems

A thesis submitted to Cardiff University

for the degree of

Doctor of Philosophy

by

Jérôme Corre

Intelligent Systems Laboratory
Manufacturing Engineering Centre
School of Engineering
Cardiff University
United Kingdom

ProQuest LLC
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106-1346

Abstract

The work presented in this thesis relates to one of the major ongoing problems in robotics: Developing control architectures for cooperation in Multi Robot Systems (MRS). It has been widely accepted that Embodiment is a prime requirement for Robotics. However, in the case of MRS research, two major shortfalls were identified. First, it was highlighted that no effort had been made into research platforms for Embodied MRS. Second, it was also observed that, generally, the more units in an MRS the lower their capabilities and as a result the poorer their degree of embodiment. These two issues were addressed separately.

Firstly, a novel concept for MRS development platform named 'Re-embodiment' is presented. Re-embodiment aims to facilitate research on control systems for MRS by minimising the effort required to ensure that the robots remain embodied and situated. Using Re-embodiment, researchers can implement and test largely different control algorithms at virtually the same time on large fleets of robots.

Secondly, an innovative mono vision distance measurement algorithm is presented. The intention is to provide a cheap, yet information rich, sensory input that can be realistically implemented on large fleet of robots. After a 'one off' calibration of the image sensor, distances from the robot to objects in its environment can be estimated from single frames.

Acknowledgements

For if none of the leaves of this thesis could be remembered, save one, it should be the acknowledgement. Although a thesis is the fruit of an individual's toil, it is only achievable through the support and faith of others. Many were those who assisted, inspired and trusted me, none shall go unmentioned.

First of all, I would like to sincerely thank my supervisor, Professor D. T. Pham, for his constant support during the past five years. The freedom he gave me allowed my intuitions and imagination to turn into ideas. His guidance turned these ideas into constructive work. His criticism steered this work into what I hope shall be worthwhile to others.

Many thanks go to all current and former members at the Manufacturing Engineering Centre (MEC), for providing a friendly work environment. My gratitude goes in particular to my fellow PhD students, for all our valuable discussions.

Kind thanks to all the technical staff, especially Paul Farrugia and Dave England, for their brilliant ingenuity and, on occasion, for their inexhaustible knowledge of classic Citroën engine.

Countless thanks to all my friends and family who helped me come out through this, without permanent head damage. My deepest appreciation goes to my parents and my brother for their support and advice throughout my life. Thank you to Julien, Manu and Sam for all the good times while I was not

working on this thesis. Deepest thanks go to Gavin Powell for teaching me how not to chat up women and subsequently introducing me to Abigail. Big thanks to Sam Barlow for being a great after hour culinary critic partner, around Cardiff's joints. Warm thanks to the rest of the party team: Michaël, Dave, Mel, Sue, Joanne, Gillian, David, Sajan, Jo, Anna, Lamlam, Jeremy and all the others.

Last but not least, I would like to express my innermost gratitude to my wife, Abigail, for her moral support throughout my research, for all the joy she fills me with every day, all the jolly good party time and for the year to come...And for those about to rock, we salute you.

Thank you all.

Contents

List of Figures

List of Tables

Notations

α_c ..... The skew coefficient of a camera. Defines the angle between the x and y axes of the pixels, such that $\alpha_c = 0$ when x and y are perpendicular.

B_n ..... A body with identification number or name n.

F ..... The focal plane, perpendicular to the focal axis, and such that $O \in F$ where O is the focal point. Also, parallel and offset from to the retinal plane R by a length f (the focal length.)

f ..... The focal length of a camera. The distance between the Focal plane F and the retinal plane R.

f_x ..... The focal length, of a camera, expressed in units of horizontal pixels, such that $f_x = \frac{f}{s_x}$.

f_y ..... The focal length, of a camera, expressed in units of horizontal pixels, such that $f_y = \frac{f}{s_y}$.

IDACK ..... A word used in the implementation of a Re-embodiment architecture on the SheepBots. Transmitted by a SheepBot to the workstation to acknowledge a matching *IDENT*.

IDENT ..... Identification number used in the implementation of a Re-embodiment architecture on the SheepBots. Each SheepBot has a unique *IDENT*.

$k_1 ... k_n$ ..... The radial distortion coefficients of a camera.

$L_{B_n} \times W_{B_n} \times H_{B_n}$ ..... The dimensions of the bounding volume of a body, B_n, with length L, width W and height H.

m ..... In the pinhole projection model, the vector $[U \quad V \quad S]^T$, such that the coordinate (u,v) of a point in the image reference plane are defined as $u = \frac{U}{S}$ and $v = \frac{V}{S}$ if and only if $S \neq 0$.

M ..... In the pinhole projection model, the vector $[X \quad Y \quad Z \quad 1]^T$ such that the coordinate (x,y,z) of a point in the camera reference frame are defined as $x = X$, $y = Y$ and $z = Z$.

o ..... The principal point. The point where the focal axis pierces the retinal plane R.

O ..... Optical centre or focal point, located in the focal plane. Also, centre of the standard co-ordinate system in a camera.

(o,u,v) ..... The image reference plane in the retinal plane at the principal point o and such that the vector u and v are collinear to the vector x and y of the camera reference frame.

(o_x, o_y) ..... The location of the principal point o in pixel coordinate.

(O, x, y, z) ..... The standard coordinate system of the camera, or camera reference frame. Located at he focal point O and with the z axis perpendicular to the focal plane F.

p_1, p_2 ..... The tangential distortion coefficients of a camera.

P ..... The projection matrix, in a pinhole camera model $m = PM$. Also defined a $P = P_{\text{int}} P_{ext}$.

P_{ext} ..... The extrinsic parameters of a camera, defining the location and orientation of the camera reference frame with respect to a known world reference frame. Defined as:

$$P_{ext} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix}.$$

P_{int} ..... The intrinsic parameters of a camera, used to link the pixel coordinate of an image point with the corresponding coordinate of the camera reference frame. Defined as:

$$P_{\text{int}} = \begin{bmatrix} f_x & f\alpha_c & o_x \\ 0 & f_y & o_y \\ 0 & 0 & 1 \end{bmatrix}.$$

R ..... The retinal plane, perpendicular to the focal axis, parallel and offset from to the focal plane F by a length f (the focal length.)

$R(B_n,t): \begin{bmatrix} r_{11} & r_{21} & r_{31} \\ r_{12} & r_{22} & r_{32} \\ r_{13} & r_{23} & r_{33} \end{bmatrix}$ ..... A rotation matrix describing the orientation of a body B_n at time t in a three-dimensional environment.

s_x, s_y ..... The effective horizontal and vertical pixel size of a camera.

S_n ..... A soul with identification number or name n.

t ..... A unique time along the time dimension in an environment.

$T(B_n, t) : \{x, y, z\}$ ..... A translation vector describing the location of a body B_n at time t in a three-dimensional environment.

Abbreviations and Acronyms

AI ..... Artificial Intelligence.

AS ..... Agent System.

BIOS ..... Basic Input/Output System.

BOM Robots..... Bunch Of Mobile Robots.

CAI ..... Classical Artificial Intelligence.

CPU..... Central Processing Unit.

DoF ..... Degree of Freedom.

FM..... Frequency Modulation.

FS..... File System.

GOFAI ..... "Good Old Fashion Artificial Intelligence."

GOFAIR ..... "Good Old Fashion Artificial Intelligence and Robotics paradigm."

GPS..... Global Positioning System.

I/O ..... (also IO) Input/Output.

IBM-PC..... Also known as IBM-Compatible PC. IBM-compatible Personal Computer.

IEE ..... Institution of Electrical Engineers.

IEEE..... Institute of Electrical and Electronics Engineers. (Pronounced Eye-triple-E.)

IP..... Internet Protocol.

IR ..... Infra-Red.

IrDA ..... Infrared Data Association.

ISA-Bus ..... Industry Standard Architecture Bus. Originally called Advanced Technology Bus (AT Bus).

LAN ..... Local Area Network.

LILO ..... Linux Loader.

MEC ..... Manufacturing Engineering Centre.

MIT ..... Massachusetts Institute of Technology.

MRS ..... Multi Robot Systems.

MS-DOS ..... MicroSoft Disk Operating System.

NAI ..... New Artificial Intelligence. Alternatively, Nouvelle Artificial Intelligence.

OS ..... Operating System.

PC ..... Personal Computer.

PCMCIA ..... Personal Computer Memory Card International Association. Alternatively, "People Cannot Memorise Computer Industry Acronyms."

PhD ..... Philosophy Doctor. Alternatively amongst PhD students, Permanent Head Damage or Patiently Headed Downhill.

PSU Board ..... Power Supply Unit.

R2D2 ..... Relative Robot Direction Determination.

RAM ..... Random Access Memory.

RCX Brick ..... Robot Command Explorer Brick.

RF ..... Radio Frequency.

ROM ..... Read Only Memory.

SAS ..... Software Agent System.

SMPA ..... Sense-Model-Plan-Act paradigm.

USART ..... Universal Synchronous Asynchronous Receiver Transmitter.

Chapter 1 Introduction

1.1 Preamble

Mankind has evolved into a unique species amongst the creatures on earth. Its emerging intelligence has fuelled its thirst for knowledge. In modern human civilisation, the understanding of the world has grown further than ever before, from the infinitely small to the infinitely large. This has driven people's ability to create and achieve. Yet mankind is still puzzled about man, furthermost by the understanding of his own intellect. For centuries, man has dreamed to be able to reproduce his ultimate power: intelligence. The first records of this fantasy date back to the ancient Greeks who were clearly ensnared with the idea of creating artificial living beings. This was to be the root of Artificial Intelligence and modern Robotics.

Much progress has been achieved in the period between those early attempts to the latest humanoid biped **[Honda, 2004]**, but robots are still far from having the versatility and intelligence of humans. Nowadays, Robotics is not just about creating artificial beings and mimicking intelligence. It is a converging point for many sciences. Robots are complete systems; they include mechanical, electric and electronic parts in both their sensors and actuators. The robot controller is the result of many fields of research, from psychology

to computer science. Multi Robot Systems, in particular, have emerged as being ideal platforms for the integration of all those sciences.

1.2 Motivation and Objectives

The work presented in this thesis relates to one of the major ongoing problems in robotics: Developing control architectures and cooperation in Multi Robot Systems (MRS). MRS have aroused a great deal of interest amongst Robotics researchers. As a logical extension to single-unit Robotics, MRS have emerged as a distinct field of research. Although this field has been the subject to much theoretical and experimental work for over the last two decades, no definitive solution has yet emerged.

The first effort in the work presented in this thesis was to review and assess existing work and different control paradigms designed for, or applied to, Multi Robot Systems. One of the fundamental principles of Robotics, acknowledged in almost all current research, is embodiment. Based on this review, two key points, neglected or overlooked by research until now, were identified. Each of these points was then addressed in turn.

The basic idea behind embodiment is that robots are meant to be progressing in an environment **[Brooks, 1991a; Brooks, 1991b; Anderson, 2003a]**. They must be able to both 'sense' and 'act-upon' their environment. As a result robots are complex systems. Furthermore, for the research to remain of interest to potential users, theoretical concepts have to be implemented and

tested in the real world. Yet, no research effort has concentrated on test and development platforms that facilitate embodiment.

To address this, a novel concept for MRS development platforms named 'Re-embodiment', is presented. Inspired by dualisms, soul and body are defined as the two building blocks of embedded robot systems. The proposed paradigm allows for largely different control systems to be implemented and tested, on a single hardware platform, by a number of researchers at virtually the same time. The effort required to ensure that the implemented system retains a good level of embodiment is minimised.

The first and most apparent appeal of multi robot systems is that increasing the number of units, performing the same task, should increase the overall performance. Unfortunately, larger fleets tend to consist of robots with lower capabilities, usually at both the computational and sensory-motory levels. One may assume that cost restrictions only allow for more computational power, better sensors and actuators on single units. Yet to remain as embodied as possible, robots must have good sensory and motory abilities.

This problem is addressed by the development of a simple, cheap, yet information rich, sensory input. A novel mono vision distance measurement method is proposed. Images are captured from an inexpensive camera. The computational requirements, especially those for image analysis, remain relatively low. After a one-off calibration, to derive intrinsic and extrinsic

parameters, distance can be measured between the robot and objects in the environment, from single image frames.

1.3 Outline of Thesis

The preceding section outlined the motivation and objectives of the work presented here. This thesis contains a further three main chapters and a conclusion.

In the second chapter, a literature review of theoretical and experimental work related to cooperation and control of MRS is presented with a special emphasis on embodiment. Two main weaknesses in the research field are highlighted. They are then addressed, one by one, in the subsequent two chapters. Each of these chapters is self-contained and includes a specific literature review, a material and methodology subsection where applicable, a theoretical section, as well as discrete results, discussions and conclusions.

Chapter Three introduces the novel "Re-Embodiment" concept for MRS development platforms. A specific literature review presents previous works. Then, the robots used for tests and experiments are described. Subsequently, the architecture is formally introduced through a series of axioms. Possible applications are highlighted, putting forward advantages. Simple experiments demonstrate the abilities of such a system. Finally, two applications of the architecture are presented, firstly, on a fleet of six PC-based robots and secondly, a larger fleet of small microcontroller-based robots.

The fourth chapter presents the innovative mono vision distance measurement algorithm. A literature review presents existing related sensing methods. Then the underlying mathematical model is detailed together with a theoretical accuracy analysis. An implementation on a cheap webcam is presented together with experimental accuracy results. Finally an application example in a simulated environment is demonstrated.

The fifth chapter concludes the work and highlights the contributions of the research presented in this thesis. A number of suggestions for future work are put forward.

Chapter 2 Literature Review

2.1 The Roots of Robotics

It is in ancient Greece that one may find the earliest record of automatons, which were to become the roots of modern robotics. In Iliad **[Homer, 2002]**, Homer describes several fictional artificial beings such as the 'tripods' and the 'golden maids of Hephaestus'. The Statesman, Philosopher and major Pythagorean mathematician Archytas of Tarentum built a wooden pigeon that it is said could flap its wings and fly **[Encyclopædia Britannica, 2004a]**. In China during the Han dynasty, around 300BC, craftsmen built a mechanical orchestra **[Encyclopædia Britannica, 2004b]**. Following the decline of Greece and Rome, interest in automata was rekindled in Mesopotamia. Abu al-Razzaz Jazari was an engineer working for the Artukid sultan Nasir ad-Din Mahmud (1200-1222). He built automata the technical perfection of which pleased the sultan, such as a clock made with a palanquin with characters and dragoons, set on an elephant and his mahout **[al-Jazari, 1975]**.

In the 18th century clockwork automata, such as the 'The flute player' and 'The duck' created by Jacques de Vaucanson **[Encyclopædia Britannica, 2004c]**, re-introduced the idea of 'artificial beings' who could do what humans can do.

In 1921, the Czech author Karel Capek coined the term Robot in his best known work, the play "Rossum's Universal Robots" (R.U.R.), which featured machines created to simulate human beings **[Capek, 1921]**. The term is believed to be derived from the Czech word for 'forced labour' or serf. R.U.R's theme, in part, was the dehumanization of man in a technological civilization. Surprisingly, Capek's robots were not mechanical in nature but were created through chemical means.

The term 'Robotics' referring to the study and use of robots was introduced and first used by the Russian-born American scientist and writer Isaac Asimov. He wrote prodigiously on a wide variety of subjects and was best known for his many works of science fiction. The word 'Robotics' was first used in Runaround, a short story published in 1942 **[Asimov, 1942]**. 'I, Robot', a collection of several of these stories in which Asimov proposed his three 'Laws of Robotics', was published in 1950 **[Asimov, 1950]**. He later added a 'zeroth law'.

> *"Zeroth Law: A robot may not injure humanity, or, through inaction, allow humanity to come to harm.*
>
> *First Law: A robot may not injure a human being, or, through inaction, allow a human being to come to harm, unless this would violate a higher order law.*

Second Law: A robot must obey orders given it by human beings, except where such orders would conflict with a higher order law.

Third Law: A robot must protect its own existence as long as such protection does not conflict with a higher order law. "

After the second world war, together with the appearance of the first computers such as the 'Manchester Mark I' **[Encyclopædia Britannica, 2004d]**, the first modern mobile robots made their appearance. Those were to renew interests in mobile robots, arouse imaginations and ambitions in modern society to finally bring robotics to the forefront of research and technology.

2.2 Emergence of Modern Mobile Robots

In the late 1940's Walter carried out pioneering research with autonomous mobile robots, 'The Machina Speculatrix', at the Burden Neurological Institute in Bristol **[Walter, 1950; Walter, 1951; Walter, 1963]**. As part of his quest to model brain functions, he was attempting to research the basis of reflex actions and tested his theory on complex behaviour arising from neural interconnections, (see Figure 2-1). The highly successful experiments with his tortoises, robots 'Elsie' (Light sensitive with Internal and External stability) and 'Elmer' (ELectro MEchanical Robot), are a landmark of robotics research **[Holland, 1997; New Scientist, 1998]**. It is fascinating to see that Walter's ideas and experiments included, at least, three of the main components of modern robotics research: Reactive control, behaviour arising

from neural interconnections and the use of a system comprising several robots. Walter's robots were using reactive control by means of sensory stimuli, in this case light, to directly influence actions. Such a control architecture was not used throughout the 'Classical' Artificial Intelligence era of robotics. It is not until the 1980s that Braitenberg **[Braitenberg, 1984]** developed Walter's experiments. A few years later, Brooks extended Walter's architecture to the so-called behavioural approach, as discussed later. The theory stipulating that behaviour arises from neural interconnections was the basis of extensive research much of which was to be applied to robotics, such as neural networks. Finally, Walter employed in his experiment two robots. It is not apparent that he was purposely researching properties arising from the use of multiple robots. However there is evidence that 'Elsie' and 'Elmer' did interact during experiments (see Figure 2-5). This may have inspired others to investigate robot interaction.

Lately an international workshop titled "Biologically-Inspired Robotics: The Legacy of W. Grey Walter" was organised in Bristol. The workshop commemorated the 25th anniversary of Walter's death. It focussed on his pioneering work in cybernetics and artificial life, on the many important and exciting developments in the field since his death, on recent trends in biologically-inspired robotics, and the potential of the field for the future **[Holland, 2003]**.

In 1964 at John Hopkins University Applied Physics Lab, 'Beasts' were presented **[Moravec, 1998]**. They were able to wander white corridors using

ultrasound sensors, until their batteries ran low. Then they would seek black wall outlets with special photocell optics, and plug themselves in by feel with their special recharging arm (see Figure 2-2). After recharging, they would resume patrolling. The Beasts were much more complex than Elsie or Elmer, and demonstrated some deliberate behaviour.

From 1966 through 1972, Shakey, so-called because of its jerky motion, was developed at the Stanford Research Institute (The SRI's Artificial Intelligence Center nowadays) **[Fikes *et al.*, 1972; Nilsson, 1984]**. It had a TV camera, a triangulating range finder, and bump sensors, and was connected to DEC PDP-10 and PDP-15 computers via radio and video links (see Figure 2-3). Shakey used programs for perception, world-modelling, and acting. Low-level action routines took care of simple moving, turning, and route planning. Intermediate level actions strung the low level ones together in ways that robustly accomplished more complex tasks. The highest level programs could make and execute plans to achieve goals given to the robot by a user. The system also generalized and saved these plans for possible future use. Shakey was the first mobile robot to reason about its actions and has had a substantial legacy and influence on present-day artificial intelligence and robotics. It now resides in the Computer History Museum in Mountain View, CA.

In the early 1970s, the Stanford Cart was developed at Stanford University **[Moravec, 1983]**. It was first designed to follow a white line but could be remotely operated as well. A prototype vision system was added in 1979 (see Figure 2-4). This enabled it to cross a thirty-meter room dotted with obstacles but travel time was a lengthy five hours!

From then on research in Artificial Intelligence and robotics became more and more widespread and as a result the number of research projects and commercial platforms being developed increased.

Even if, nowadays, the general perception of robots still undoubtedly overshadows the reality of their abilities, Robotics remains at the forefront of technology. It is a converging point to many sciences and provides an ideal platform to integrate research from different fields into complete systems.

2.3 Robot Control

Robotics researchers are faced with the task of engineering machines that gather information about their environment via sensors and effect action via actuators. The link between the sensor data and the actuator effect is the control. To understand the evolution of the underlying ideas behind robots control, one has to go back to eighteenth century dualism.

Figure 2-1. Walter Working on One of his Tortoises [Walter, 1950; Walter, 1951]

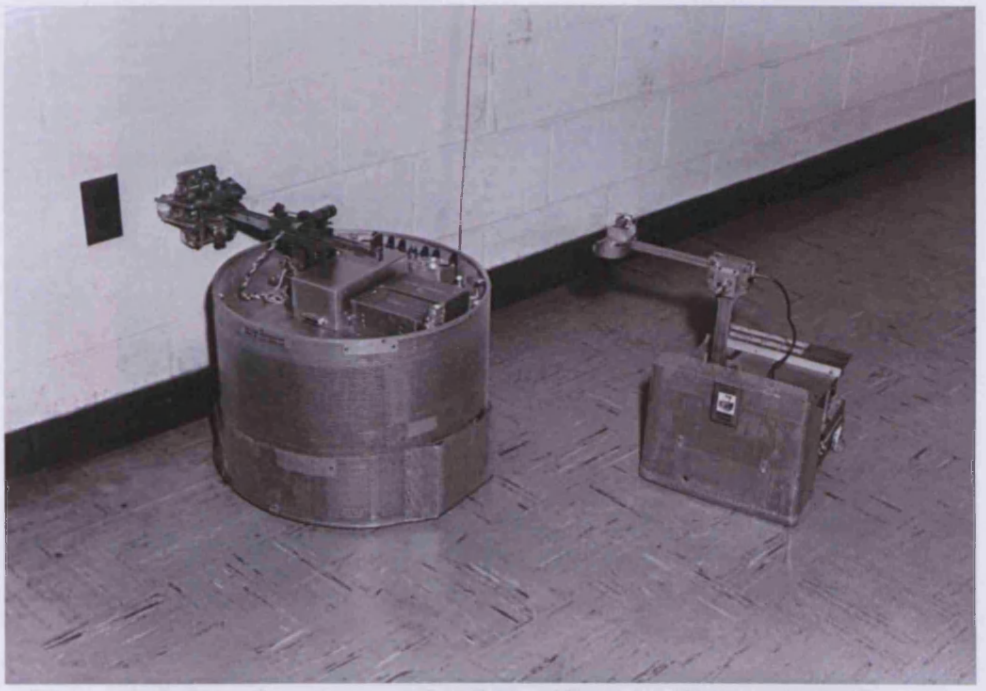


Figure 2-2. Two Versions of the Johns Hopkins University's 'Beasts' [Moravec, 1998]

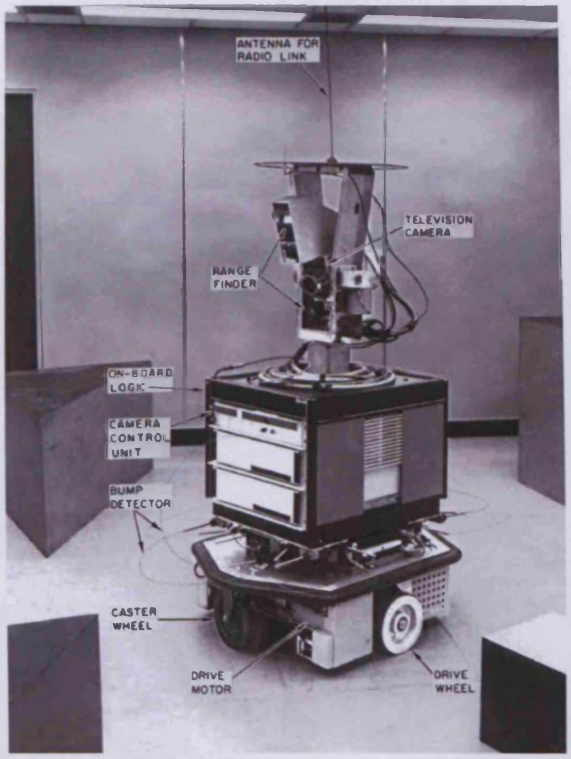


Figure 2-3. Shakey, So-Called Because of its Jerky Motion [Fikes *et al.*, 1972]

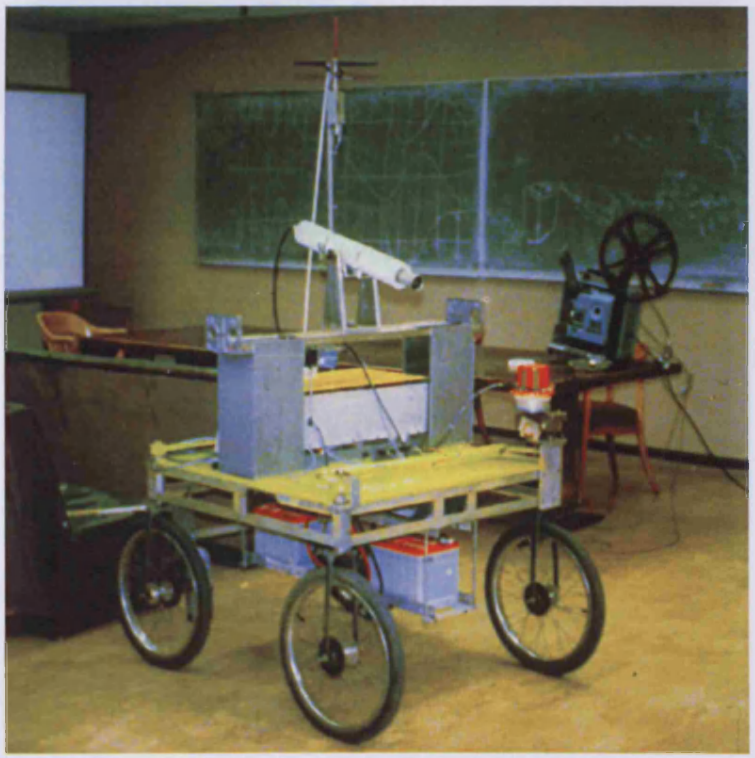


Figure 2-4. The Stanford Cart with its Prototype Vision System [Moravec, 1983]

Figure 2-5. 'Elsie' and 'Elmer': Was This the First MRS System?[1]

[1] In this experiment, Walter observed the strange movements of the 'dance' of two turtles, each with a light sensor and a light. Was this the first experiment involving interacting robots?

2.3.1 The Nature of Dualism

The term 'dualist' was first coined by Thomas Hyde in the early 18th century **[Hyde, 1700]** to describe religious systems that consist of God and the devil as two co-eternal principles. However, it is Christian von Wolff who first used the term 'dualism' in 1734 **[von Wolff, 1734]**. He also introduced both terms into philosophical discourse. As a metaphysical theory, dualism states that the world is made up of two irreducible elements. This includes distinctions such as mind and body, good and evil or phenomena and noumena. Here, it is the mind-body distinction, which is of interest. It is also sometimes referred as the body and soul distinction.

Although this split between body and mind can be traced back to the ancient Greek philosopher Plato **[Plato, 360 BC]**, it is René Descartes who undertook the first systematic analysis of the mind body issue. In the first of his works, although published after his death, 'De Homine' **[Descartes, 1664]**, Descartes proposed a mechanism for automatic reaction in response to external events. This work also outlined the views that provided the first articulation of the mind-body interactionism. In 1641, 'Meditationes De Prima Philosophia, In Quibus Dei Existentia, & Animæ Humanæ à Corpore Distinctio Demonstrator' was published **[Descartes, 1641]**. In this work, Descartes presented the first extended discussion of the metaphysical distinction between mind and body. He reasoned that the world was divided into two domains: the non-physical - the mind and the physical - the body. According to his views, mind (or soul)

and body are different substances, i.e. distinct and independent types of being. Descartes also suggests that only the mind is intelligent and that the body is only an interface to the world.

Philosophers after Descartes, such as Locke **[Locke, 1690]**, Kant **[Kant, 1798]** or Müller **[Müller, 1834-40]**, refined this distinction between the qualities of the mental and the physical. The mind is active, invisible and intangible, while physical objects are inactive, visible and tangible. Most dualist theories of mind are based on Cartesian dualism. However, Descartes realised that the mind-body dualism is beset with an interaction problem. How is interaction possible between the mind and the body? For instance, how does one feel thirsty (mental event) when one's body needs a drink (physical event)?

In 1949, Ryle raised fundamental criticisms of the dualist theories. In his book 'The Concept of Mind' **[Ryle, 1949]**, he rejected dualism, which he called 'the dogma of the ghost in the machine.' Ryle argued that dualism makes 'category mistakes'. Dualist doctrines establish a polar opposition between mind and body. At the language level, the mental properties are logical negation of the physical properties. Body is extended, has location and has parts, whereas mind is not extended, has no location and has no parts. So they belong, in accordance with the concept of category, to the same logical type, given that the expression used for the description of mental events are always mere negatives of the expressions used for description of natural events. According to Ryle this implied a category mistake, as the idea of mind cannot be a substance since it is only understood in contrast to properties that are

assigned to substances. This is not the only category mistake that Ryle thinks dualists have committed. Other mistakes involve for instance mentalistic terms (such as mind, idea and pain) and the statements, which contain those terms.

Ryle's solution is to argue that the correct use of words like 'mind', 'idea' or 'pain' is in connection with human behaviour. For instance, having an idea means behaving in certain ways or being disposed to behave in such ways, by saying certain things, having a certain expression on one's face etc. It is not having something intangible and invisible floating through one's head. Ryle's Theory is an attempt to explain away the mind. Although it is permissible to talk about doing things with one's mind, for Ryle this means something quite different from what it would do Descartes. Having a mind just means that one is disposed to behave in certain ways, i.e. if circumstances were such and such, one would do such and such. There is nothing mysterious or occult going on.

Ryle's theory was also in turn dismissed. A number of other theories try to explain the mind such as the Mind-Brain Identity Theory, Intentionality, Functionalism and Epiphenominalism. However, no single theory has been widely accepted and the theory of mind is still subject to many debates.

2.3.2 G.O.F.A.I

Descartes has been referred to as the father of modern philosophy **[Gaukroger, 1995]** following his study of the body and mind **[Descartes, 1641]**. He aimed to show that the body is distinct from the mind and that intelligence was an attribute of the spirit only.

Traditionally, the various branches of Cognitive Science also viewed the mind as an abstract information processor, whose connections to the outside world were of little theoretical importance. Sensory and motor systems, acknowledged to be objects of research in their own right, were not regarded to be relevant to the understanding of the central cognitive process. They were seen as peripheral input and output devices **[Wilson, 2002]**.

In a similar way, Artificial Intelligence (AI) research was initially trying to prove that formal symbol manipulation is both necessary and sufficient to generate intelligent behaviour **[Simon, 1957]**. This, once more, suggested that intelligence could exist without a body or an environment. This is referred to as the Classical AI approach (CAI) or as dubbed by Haugeland, the 'Good Old Fashion AI' (GOFAI) **[Haugeland, 1985]**. Classical AI aimed to model human intelligent thought and showed potential for accomplishing complex tasks. Various problems were examined like playing chess or solving puzzles. Each program had a specific problem domain. The key methods to solve these problems were 'symbolic representation' and 'symbol manipulation engines'. By using these methods, great achievements were attained, such as Deep

Blue's victory over chess grand master Gary Kasparov in 1997 **[Morris, 1997]**. However, these achievements have only been made in limited domains, referred to in the field as 'microworlds'.

GOFAI as applied to robotics is also called Good Old Fashion Artificial Intelligence and Robotics (GOFAIR) **[Mackworth, 1992; Sahota and Mackworth, 1994]**. When researchers first started developing control architectures for mobile robots, the natural approach was to use the symbolic manipulation systems developed by AI. The 'only' other requirement then was to develop a perception system to abstract sensory information into symbols and a mechanism for turning symbols into actuator commands. These control paradigms are referred as the Sense-Model-Plan-Act (SMPA) framework **[Brooks, 1991a]**. Figure 2-6, shows a decomposition of the functional modules used in this type of mobile robot control system. GOFAI robots fuse the varying sensor data in order to develop some kind of world model. This world model is then used in varying degrees of reasoning and planning for action. Finally, the action is executed. These classical approaches have shaped research in Artificial Intelligence and Robotics since the late 1960's. Although they proved to be a rich source of control ideas, especially when results were interpreted by a human, problems arose when applied to autonomous robots.

For instance, for the development of Shakey **[Fikes *et al.*, 1972; Nilsson, 1984]**, several fundamental assumptions were made about the world. Firstly, there is only one agent. Secondly, the environment is static unless the agent

changes it. Thirdly, actions are discrete and carried out sequentially. Finally, the world the robot inhabits can be modelled accurately and exhaustively by the robot. Although Shakey is a landmark in the development of robotics, these assumptions proved to be too restrictive **[Sahota and Mackworth, 1994]**.

More generally, GOFAI when applied to Robotics suffered a number of limitations. Firstly, it was very sensitive to sensor noise and uncertainty. Secondly, it was computationally cumbersome and required large amount of processing power. Finally, it also tended to struggle with the real-time requirement of robotic applications. The most cited example is the Stanford Cart, which spent so much time computing the Planning stage, that its visual reasoning was disrupted by the changing shadows due to sun movement **[Moravec, 1980]**.

2.3.3 Behaviour-Based Approach

The argument that a robot would simply provide the sensors and actuators for an artificial brain became seriously flawed. The whole ethos was plagued with problems such as real-time performance, stability through sensor noise or maintaining representational model validity.

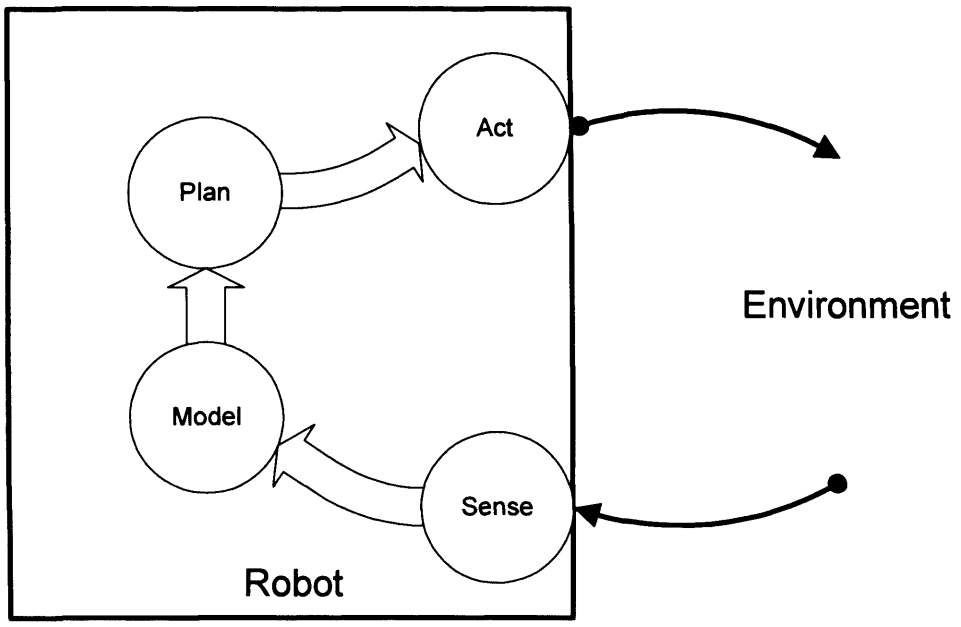


Figure 2-6. The Sense-Model-Plan-Act Paradigm of GOFAI

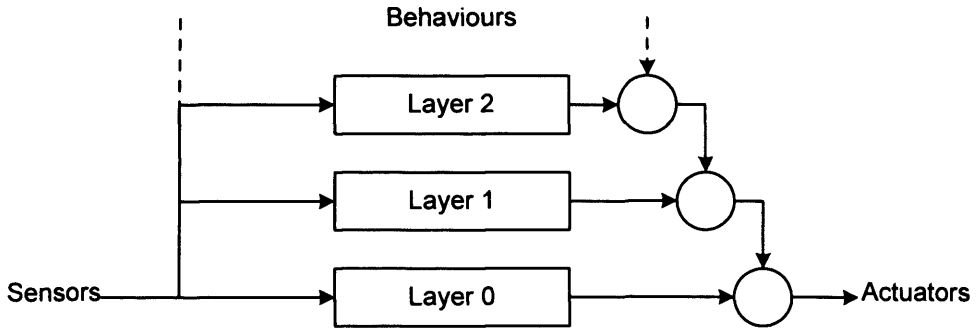


Figure 2-7. The Layered Subsumption Architecture

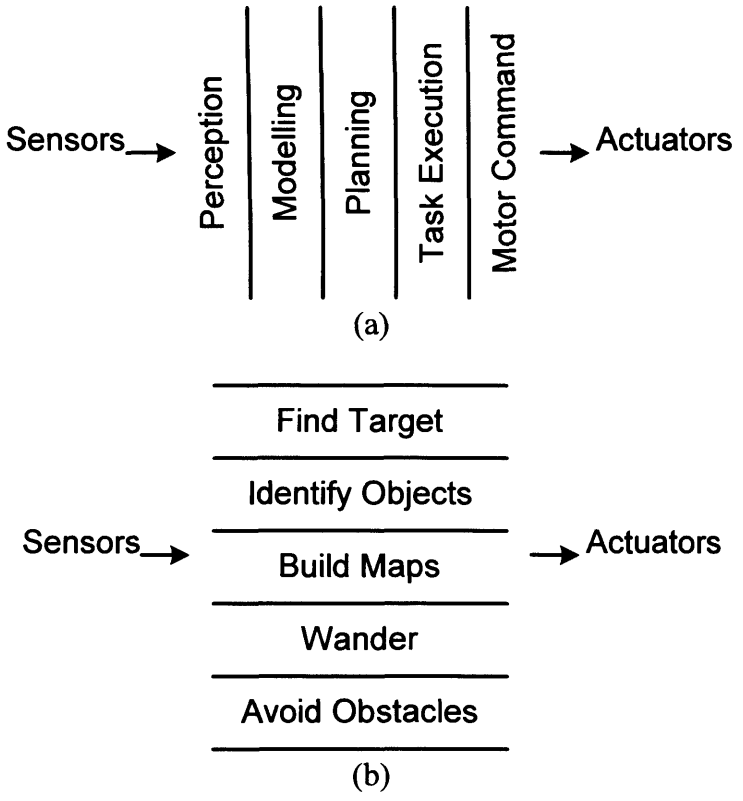


Figure 2-8. The Horizontal (Serial) Architecture of GOFAI (a) versus the Vertical (Parallel) Approach of Subsumption(b)

The inability of classical approaches to handle unconstrained interactions with the real world has led to new research in control architectures for autonomous agents. It became apparent that focussing on the interaction between the system and the environment was essential to achieve robust control for autonomous robots. This new approach was led by a series of provocative publications by Brooks presenting the 'subsumption architecture' **[Brooks, 1986; Brooks, 1991a; Brooks, 1991b]**.

Brooks popularised in Artificial Intelligence and robotics ideas that had roots in divergent branches of Philosophy, Psychology and Cognitive Science. The developmental Psychology of Jean Piaget, which emphasises the emergence of cognitive abilities out of groundwork of sensorimotor abilities **[Piaget, 1946]**. The claims of German philosopher Heidegger that man functions in his environment because he is part of it **[Dreyfus, 1991]**. In Cognitive Science, motor theories of perception such as those suggested by William James and others were reviewed by Prinz **[Prinz, 1987]**. Brooks argued that both situatedness and embodiment are 'cornerstones' for the development of Artificial Intelligence. His work formed the basis of New AI or Nouvelle AI (NAI). The contrast between Classical AI and New AI is also referred to as 'representation versus perception' **[Duffy and Joue, 2000]**.

By *"situated"*, Brooks refers to creatures or robots being embedded in the environment. They do not deal with an abstract description of the world. Instead, their behaviour is directly affected, through their sensor, by the 'here and now' of the world.

By *"embodied"*, Brooks refers to creatures or robots experiencing the world directly through a physical body. Their actions are part of a dynamic interaction with the world and have immediate effects on their own sensations **[Brooks, 1991a]**

Mataric also strongly supports Brooks view on situated and embodied Robotics **[Mataric, 2003]**. They share the view that the two terms are somehow related **[Brooks, 1991b; Brooks, 2002; Mataric, 2003]**.

Motivation for behavioural-based robotics stems from views that intelligence is something that happens in close interaction with the current situation. This in turn suggests that building a world model from sensor inputs in order to accomplish tasks is unnecessary. Instead, robots only need to process aspects of the world that are relevant to their tasks **[Brooks, 1991a; Brooks, 1991b]**. As shown in Figure 2-7, Subsumption architectures comprise built up 'layers' of 'competence'. These are arranged in a vertical structure, as opposed to the horizontal GOFAI approach (see Figure 2-8). Each layer represents a single primitive behaviour. This method is known as Subsumption because each higher level behaviour includes the lower levels as a subset. In

addition, higher levels subsume the lower levels by suppressing their output. Each new level adds to the overall competence of the robot. If each layer can be debugged before adding another, without disrupting the functioning of the lower ones, then the layers can be developed independently. This makes the design task considerably more manageable. Furthermore, these modular layers with very little cross communication, allow for behaviour to be added and removed as needed. One must bear in mind that each of these behaviours is simple when taken individually. The functionality of the system emerges from the interaction between these different primitive behaviours and the perceived environment. Taken as a whole, the system exhibits complex behaviours, seemingly making intelligent decisions about how to interact with its environment. These more complex behaviours are never explicitly designed into the robot. They simply emerge from complex interactions between the multitude of primitive behaviours designed into the system and the environment in which the robot exists. However, the designer is usually aware of the emergent behaviour, or even specifically designs to induce an emergent behaviour.

Since it requires no internal model of the environment, the behaviour-based approach to robotics was seen as the answer to the problems of GOFAI. It is ideally suited to the real-time computing necessary in mobile robotics because it offers the advantage of extremely quick processing. Although many successes have been achieved with this type of control architecture, two main problems stand out **[Brooks, 1991a; Etzioni, 1993]**. Firstly, it is not known

how well they will scale. Secondly, there exists nothing like a Turing equivalence theorem that states, at least in principle, whether these schemes can be used to accomplish anything that may be desired of them. In addition, there are no analytical tools for understanding in advance what sort of conflicts and other unexpected interactions might arise from the ways behaviours are combined using these methodologies. In practice, it proved difficult to develop behavioural systems that ensure overall goal achievement, especially with more complex goals. This led, in recent years, to a number of new hybrid approaches that attempt to encapsulate the benefits of both architectures **[Low *et al.*, 2002]**. These deliberative-reactive architectures allow the system to plan for goal achievement while maintaining immediate responsiveness to the environment.

Nevertheless, Brooks's revolutionary ideas have forever changed the fundamental principles of Robotics by enforcing the requirement for Embodiment.

2.3.4 The Generalisation of Embodiment

In recent years, the importance of the interaction between a system or robot and its environment has become widely accepted in fields such as Artificial Intelligence, Artificial Life and Embodied Cognition. It is now seen as a 'condition sine qua non' **[Pfeifer and Scheier, 2001; Ziemke, 2001a, Ziemke, 2002 #108]** for the development of intelligent behaviour. For instance, Dautenhahn stated that:

> *"Life and intelligence only develops inside a body, which is adapted to the environment which the agent is living in. Intelligence can only be studied with a complete system, embedded and coupled to its environment."* ***[Dautenhahn, 1999 p.1]***

Pfeifer *et al.* also suggest that:

> *"Intelligence cannot merely exist in the form of an abstract algorithm but requires a physical instantiation, a body"* ***[Pfeifer and Scheier, 2001 p.649]***

Wilson recently emphasised this emerging trend:

> *"The emerging viewpoint of embodied cognition holds that cognitive processes are deeply rooted in the body's interaction with the world"* ***[Wilson, 2002 p.1]***

Although embodiment and situatedness have become important concepts in many areas, the terminology has remained very unspecific. Different notions emerged as to what embodiment and situatedness might mean. It is also unclear how the ideas of embodiment and situatedness overlap or diverge **[Brooks, 2002 pp.51-52]**. Different terms related to situatedness have appeared in the literature, such as 'Situated Action' **[Suchman, 1987]**, 'Situated Cognition' **[Clancey, 1997]**, 'Situated AI' **[Husbands *et al.*, 1993]**, 'Situated Robotics' **[Brooks, 1991a; Hallam and Malcolm, 1994; Brooks, 2002; Mataric, 2003]**,

'Situated Activity' **[Hendriks-Jansen, 1996]**, 'Situated Translation' **[Risku, 2002]**. Note that this list is non-exhaustive.

The notion of Embodiment has also generated a myriad of expressions, such as 'Embodied Mind' **[Varela *et al.*, 1991; Lakoff and Johnson, 1999]**, 'Embodied Intelligence' **[Brooks, 1991a]**, 'Embodied Action' **[Varela *et al.*, 1991]**, 'Embodied Cognition' **[Clark, 1997]**, 'Embodied AI' **[Franklin, 1997]**, 'Embodied Cognitive Science' **[Pfeifer and Scheier, 2001]** and 'Embodied Evolution' **[Watson *et al.*, 1999]**.

This diversity does not only apply to the terminology itself. The notions behind the terms also diverge. Along with the interpretations put forward by Brooks **[Brooks, 1991a]**, there are also a number of other explanations. For instance, Pfeifer *et al.* state that:

> *"In artificial systems, the term [embodiment] refers to the fact that a particular agent is realised as a physical robot or as a simulated agent."* ***[Pfeifer and Scheier, 2001 p.649]***

Terada *et al.* declare that:

> *"We [Terada et al.] define the term embodiment as the extent of the agent's body, locomotive ability and its sensor. The extent means how and how much the agent's body occupies in the physical space, that is the shape and size of the agent's body. [...] In other words, the significance of existence of the object depends on the embodiment of the agent.*

*Therefore, the embodiment of the agent can be used to represent the relationship among its behaviour and environment." **[Terada et al., 2001 p.2]***

These differing notions of embodiment and situatedness have led to different opinions regarding the required conditions for a system to be considered embodied, or situated, in its environment. Once more, this led to different terminology such as, 'Situated Embodiment' **[Zlatev, 1997; Zlatev, 2003]**, 'Natural Embodiment' **[Ziemke, 1999]**, 'Naturalistic Embodiment' **[Zlatev, 2001]**, 'Mechanistic Embodiment' **[Sharkey and Ziemke, 2001]**, 'Phenomenal Embodiment' **[Sharkey and Ziemke, 2001]**, 'Social Embodiment' **[Barsalou *et al.*, 2003]**, 'Social Embeddedness' **[Dautenhahn *et al.*, 2002]** 'Physical Embodiment' **[Ziemke, 2001a]**, 'Organismoid Embodiment' **[Ziemke, 2001a]** and 'Organismic Embodiment' **[Ziemke, 2001a]**.

One of the main debates is whether a physical body is required to achieve interaction with the environment. It is even unclear whether the environment itself has to be physical or not. Some, such as Brooks, Mataric or Pfeifer **[Brooks, 1991a; Pfeifer and Scheier, 2001; Brooks, 2002; Mataric, 2003]** argue that a physical body is required in a physical world. Brooks states that:

"At each step we should build complete intelligent systems that we let loose in the real world with real sensing and action" ***[Brooks, 1991b p.140]***.

Others argue that a physical body is not required **[Oka *et al.*, 2001]**. Etzioni goes as far as replying directly to Brooks and argues that software environments such as operating systems or databases are a valid substrate for intelligent agent research **[Etzioni, 1993]**. Franklin uses autonomous software agents as cognitive models to generate testable hypotheses about human cognition **[Franklin, 1997; Franklin and Graesser, 2001]**. He argues that:

"Software systems with no body in the usual physical sense can be intelligent. But they must be embodied in the situated sense of being autonomous agents structurally coupled with their environment" ***[Franklin, 1997 p.1]***

Kushmerick demonstrates that embodiment in a software world, such as databases or the Internet, fits with the notion of interaction between an agent and its environment **[Kushmerick, 1997]**.

Another unclear line of reasoning that has filtered into the literature revolves around the manner by which a system interacts with its environment. For instance, Werger and Mataric make a distinction between different levels of interaction **[Werger and Mataric, 1999]**. There exist systems where agents simply react to their environment. There are also systems where agents affect

each others behaviours through more than mere interference, in applications such as flocking. Furthermore, there are systems that make use of environmental modification, for example, territorial marking or pheromone trails. Quick *et al.* highlight the fact that some robots have more sensory capabilities than others, thus suggesting that the extent of the embodiment has an impact on the capabilities of the system in an environment **[Quick and Dautenhahn, 1999]**. In another publication, they use the following example:

> *"In terms of relationships with the social world (i.e. robot–human interactions), the Aibo robot dog (Sony) has a greater range of interactive skills than for example a Khepera robot (K-Team)."* ***[Dautenhahn et al., 2002 p.399]***

Dautenhahn *et al.* mention that robots have differing relationships with the world. They mention that a mobile robot with translational and rotational capabilities in two dimensions (i.e. two degree of freedom) would have less capability to interact with the world than a robot that also possesses an arm with an extra five degree of freedom **[Dautenhahn, 1997; Dautenhahn *et al.*, 2002]**.

Another point still being investigated is representation **[Keijzer, 2002; Robbins, 2002]**.

A recent paper by Anderson provides a review of publications, notions and points of view related to embodiment **[Anderson, 2003a]**. Anderson attempts to present the current attitude and goals of the research concerned with embodiment. However, there is still much debate **[Anderson, 2003b; Chrisley, 2003]**.

This lack of uniformity has been acknowledged in the literature. For instance, Wilson indicated that this diversity is cause for concern, by stating that:

> *"While this general approach [of embodied and/or situated cognition] is enjoying increasingly broad support, there is in fact a great deal of diversity in the claims involved and the degree of controversy they attract. If the term 'Embodied Cognition' is to retain meaningful use, we need to disentangle and evaluate these diverse claims."* ***[Wilson, 2002 p.2]***

Quick *et al.* attempt to provide a formal definition embracing all the above notions. They first look at the underlying principles, common to all those different 'situated/embodied/embedded/interactive' theories, as Ziemke calls them **[Ziemke, 2001a]**. Quick *et al.* identified the sensorimotor dynamics between a system and its environment as a common principle. These dynamics were particularly emphasised by Beer **[Beer, 1995]**. They use the term 'structural coupling' to denote these interactive dynamics **[Quick and**

Dautenhahn, 1999; Quick *et al.*, 1999a; Quick *et al.*, 1999b]. 'structural coupling' was first used in this sense by Maturana and Verela **[Maturana and Varela, 1980]**. Quick *et al.* then use the more common terms 'embodiment' and 'embodied' to refer to systems which have structural coupling with their environment.

Quick *et al.* then put forward a minimal definition of embodiment **[Quick and Dautenhahn, 1999; Quick *et al.*, 1999a; Quick *et al.*, 1999b]**.

> *"A system **X** is embodied in an environment **E** if perturbatory channels exist between the two. That is, **X** is embodied in **E** if for every time **t** at which both **X** and **E** exist, some subset of **E**'s possible states with respect to **X** have the capacity to perturb **X**'s state, and some subset of **X**'s possible states with respect to **E** have the capacity to perturb **E**'s state."*

Through this definition, Quick *et al.* clarify that embodiment is not solely a feature of a system in an environment but is grounded in the relationship between the two.

This definition, by being minimal, embraces all of the divergent notions that appeared in the literature. Quick *et al.* do point out that this definition is minimal, and does not rule anything out, on the basis of higher theoretical situations, such as 'belief' or 'intention'. They give the interesting example of a granite outcrop in the Antarctic tundra; this illustration is also examined by Ziemke **[Ziemke, 2001b]**. The outcrop (X) in the tundra (E) is persistently

perturbed by the wind, and in turn perturbs the air currents' flow. According to the definition of embodiment by Quick *et al.*, the outcrop is an embodied system in the tundra. Ziemke argues that this definition does not make a distinction between cognitive and non cognitive systems by saying that:

> *"Certainly not many cognitive scientists would actually consider this [outcrop in the tundra,] an example of embodied cognition."* ***[Ziemke, 2001b p.2]***

However, he himself does not provide any form of definition to distinguish between systems that are cognitive and systems that are not. The definition by Quick *et al.* does however, to some extent, allow for such a distinction. The definition contains variables such as the number of possible states and the scope for their perturbation (see **[Quick *et al.*, 1999a]** for further details). These variables allow for some quantification. It could be argued that although the definition provides the opportunity to explicitly quantify embodiment, it does not yet do so with any particular metric. Furthermore, as Quick *et al.* point out **[Quick and Dautenhahn, 1999]** there is still plenty of scope for discussion on exactly how coupling occurs and what phenomena are made possible as the result of coupling. Finally, this definition of embodiment is not plagued with material constraints; in other words, a system may be embodied as per the definition without having a physical body. This formally opens the notion of embodiment to domains such as software.

In this thesis, the definition of embodiment by Quick *et al.* will be regarded as valid for describing the structural coupling between a system and its environment. It will also be regarded as providing a formal term, i.e. embodiment, for the different expressions, such as situatedness, interaction, etc that have previously been mention in the literature **[Brooks, 1991a; Ziemke, 2001b; Ziemke, 2001a]**.

2.4 Summary

This chapter has traced the development of the field of robotics, from it ancient roots to the present days. The chapter has then focussed on the current situatedness embodiment debate in robotic research

Chapter 3 "Re-Embodiment" a Novel Development Architecture Concept for Multi Robots Systems Research

3.1 Preamble

One of the basic requirements for Artificial Intelligence (AI) and robotics research is a hardware platform upon which to test and validate new theories or algorithms. Simulation provides little insight into the robustness and applicability of theory to the real world. This is widely accepted and understood at all levels in the robotics community **[Mondada *et al.*, 1993; Lichtensteiger and Ralf, 2000; Kellis, 2002]**. This alleged importance of embodiment in Robotics was first advocated by Brooks **[Brooks, 1991a; Brooks, 1991b]**. In this thesis, it is further suggested that the physical validation of algorithms should not be undertaken in engineered environments, but rather in real, dynamic settings. In other words, testing and validation of an algorithm in an 'arena' with artificial obstacles will have less significance than one conducted directly in a typically cluttered robotics laboratory.

3.2 Review of Related Work

The process by which an algorithm can be implemented and tested on a hardware platform is of utmost importance. For a research exercise to be

efficient, or in any way optimised, the implementation and testing process must be carefully thought through. This is particularly true in Robotics. Robots, being complex systems, consist of mechanical, electric, electronic and control elements. All those considerations have to be incorporated into the design and properly integrated as a whole. The problem is even more critical in multi-robot research. The number of robots together with their potential heterogeneity rapidly complicates the implementation. Furthermore, and often because of cost, several researchers may have to test largely different algorithms and theories on a common hardware platform, at the same time. This demonstrates the need for flexible platforms.

Despite this need, there have been very few publications concentrating on the creation of 'development architectures'. 'Experimental Robotics' by Wilberg and Siegberg is a rare example **[Wilberg and Siegberg, 1998]**. Software packages are available for the rapid prototyping of robot systems, such as SYMOFROS and RT-Lab **[Lambert *et al.*, 2001]**. These allow for quick modelling and simulation of robot systems, but do not provide the functionality required to test control algorithms on hardware platforms.

Commercial hardware platforms for robotics research are widely available, as single units. For instance the Pioneer mobile robot and its related products, currently commercialised by Active Media Robotics, have been extensively used around the world over the last few years **[Rus *et al.*, 1996; Gerkey and Mataric, 2002]**. Mondada and Franzi designed the very popular Khepera, a small modular mobile robot **[Mondada *et al.*, 1993]**. They also co-

founded the Swiss company K-Team S.A. which produces it commercially. Recently, Sony has released Open-R, an architecture to aid development based on the well-known Aibo entertainment robot **[Fujita and Kageyama, 1997; Fujita *et al.*, 1997,Gutmann, 2003 #123]**. The 'Lego® MindStorm' kit has proved very popular to various group, from school pupils to researchers **[Martin, 1996; Levesque and Pagnucco, 2000]**. It combines standard Lego® building blocks and a controller, the 'Robot Command Explorer' (RCX) brick, based on MIT's programmable brick **[Resnick *et al.*, 1996]**. It provides a cheap flexible platform for robot hardware design and prototyping, together with an easy-to-use controller **[Reshko *et al.*, 2002]**.

Webots **[Michel, 2003]**, a robot simulation package commercialised by Cyberbotics, attempts to fill the gap between simulation and hardware validation. It provides modelling facilities based on VRML and simulation facilities for control algorithms coded in C or Java. It also has the ability to cross-compile and transfer control programs to a Khepera robot using a serial port interface cable. However, there are still a number of limitations, such as the coding language of the source program, the version of the Khepera platform, and the physical connection required between the robot and a personal computer (PC) for the download.

There seems to be a lack of systems facilitating development and validation of algorithms for multi-robot research. One could have a fleet of single commercial units, but would rapidly find it difficult to manage as the number of robot grows.

One of the first assignments of the research presented in this thesis was to design a hardware platform based on a fleet of six existing robots know as the "Bunch of Mobile" (BOM) robots, **[Beutler, 1998]**. The platform had to be designed to ease the implementation and testing of control algorithms for multi robot systems. Although only six robots were to be used at first, expandability was a prime objective. The system was also expected to be easily usable by several researchers, testing largely different things, at the same time. Together with the development of this hardware platform architecture, several improvements were made to the individual robots **[Corre, 2001]**.

This chapter first briefly describes the BOM robots as they were at the beginning of this project, and the improvements made. Then, the novel concept of the development architecture is explained, highlighting its usability for multi-robot research. The advantages, drawbacks, and possible improvements to the architecture are discussed. Possible applications are presented. Subsequently, the implementation of the development architecture on a fleet of six PC-based robots is documented, highlighting its flexibility through an example application. Finally, a second implementation, on a larger fleet of micro-controller based robots, is described.

3.3 The BOM Robots

3.3.1 The BOM Project

The BOM project started back in 1996. It was a programme of work conducted by Beutler and seven visiting researchers **[Beutler, 1998]**. The original objective was to develop a research platform for the development of control algorithms for multi-robot systems. It was to include both software and hardware facilities.

> *"The aim of this project was the prototyping of an expandable, IBM-compatible-PC based low-cost mobile robot capable of co-operating with other robots. The hardware price limit was set to £500. Furthermore, all hardware modules developed had to follow the PC/104 standard that embodies a physically different version of the ISA-Bus.*
>
> *Further, software structures had to be developed to enable users to develop their own control algorithms. These algorithms have to be tested in simulation environments but also directly on the mobiles."* **[Beutler, 1998 p.1]**

To ensure that the BOM robots remain an efficient and powerful research platform, especially when several users are developing and testing algorithms simultaneously, several improvements were needed. The following section

gives a brief description of the original hardware of the BOM robots based on the information given in **[Corre, 2001]**.

3.3.2 The Original BOM Robots

The hardware-related part of the BOM project aimed to create several mobile robots **[Beutler, 1998]**. During the concept design stage of these robots, the following requirements were identified:

- Expandability - the system must allow future expansions of the hardware.
- Compatibility - the robot hardware must be compatible with commercially available standard components.
- Low-cost - the material cost of a single mobile must be below £500.
- Easy to program - standard software development tools must be applicable.

These demands led to the choice of an inexpensive, standard Intel processor (80386) computer motherboard as the base of the system. Furthermore, these IBM-PC compatible motherboards fulfil the need to support user-friendly programming environments.

Compatibility considerations led to the decision to build all hardware components around the ISA-Bus standard **[IEEE, 1983]**. This standard is only concerned with electrical connections (e.g. pin layout, signal timing). The physical realisation of this bus system in IBM-compatible computers is however demanding in space. Thus, the decision was made to use the PC/104 version of the ISA-Bus. This version is only a physical re-definition of the ISA-bus connectors. The PC/104 standard **[Consortium, 2001]** was first introduced by AMPRO in 1987 and is now widely accepted among manufacturers and users of embedded systems. This is reflected in a variety of products, e.g. PC-motherboards, PCMCIA modules, network cards, GPS cards, I/O-boards and memory extension modules. Physically, this new version of the ISA-bus provides a stackable design, i.e. all boards are electrically connected and physically supported by a special 'piggyback' connector. This leads to a mechanically robust and compact realisation of the embedded system. Therefore, the BOM robots are fully compatible and extendable for future developments, or user specific requirements.

The control hardware of each robot is based around a single board PC. Most peripheral hardware is connected to this board via the PC/104 bus (Figure 3-6).

Communication was based on a Frequency Modulation (FM) radio card connected directly to the RS232 serial port of the robot's on-board PC (see Figure 3-2). The Relative Robot Direction Determination (R2D2) module

provided orientation information among radio transmitting and receiving mobiles (Figure 3-2). During the transmission of a radio message, the transmitting mobile switches on an infrared (IR) light, the direction of which can be detected by all receiving mobiles using circular mounted IR sensors **[Schomaker, 1996; Djakov, 1997]**. All IR-sensors and emitters are physically separated from the amplification and A/D converter stage, and are mounted on a sensor tower to facilitate transmission in all directions (Figure 3-1).

The stepper motor drive board supports two independent stepper motors, each of which is driven with separate control signals. The signals allow for change in direction and variable step frequencies to control speed of the wheels. The interface and motor control board generates the control signals. It also boasts six counters, eight lines for digital inputs, eight lines for digital outputs and one interrupt signal, all available for extra hardware (Figure 3-3).

The distance measurement board provides information about distances around the mobile robot as well as reporting collisions with other obstacles **[Liedtke, 1997]** (Figure 3-4). This board drives six piezo-electric-based ultrasonic sensors for range measurements up to seven metres. Four switches are also used to detect eventual collisions.

The on-board power supply board manages and converts the battery voltage (+12V) into all the required voltages (+5V, -5V, +12V), for all electronic components. Furthermore, it indicates if the +5V voltage is outside an adjustable voltage band, e.g. a flat battery or a short circuit.

The chassis of the BOM robots (Figure 3-5) encloses the stepper motors as well as two gearboxes. These gearboxes reduce the stepper motor's speed with a ratio of two to one. In conjunction with its gearbox and stepper motors, the mobile is designed to pull heavy loads (up to 20 kg at a speed of 0.15 ms^{-1}, assuming no slippage of wheels) **[Cavaliere, 1996]**. The battery can be easily taken out and replaced with another to enable near continuous operation. The motor drive card and the power supply board are secured to the chassis.

3.3.3 Notable Improvements to the Robots

Several improvements were made to each individual robot (Figure 3-8). Some were very straightforward, such as securing the main PC board and the PC/104 stack to the chassis, tidying and securing the cable connections between different components (Figure 3-9) or fitting and interfacing a camera (Figure 3-10). Other improvements are of more notable interest. Although not the subject of detailed study in this research, they are mentioned since they provide improvements or solutions to common problems in robotics.

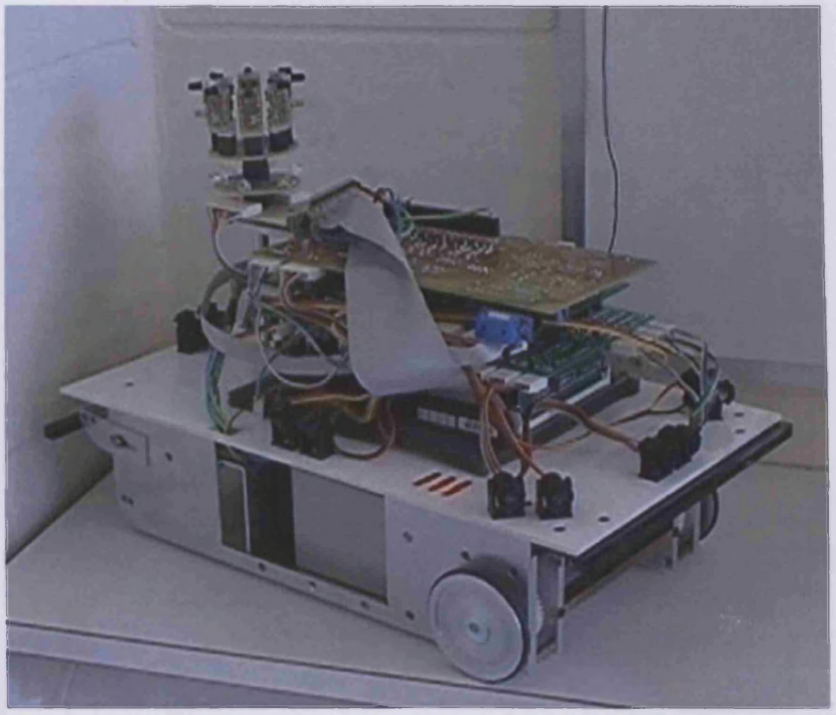


Figure 3-1. An Original BOM Robot.

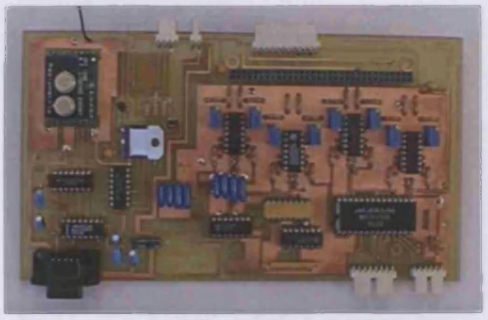


Figure 3-2. The R2D2 and Radio Board.

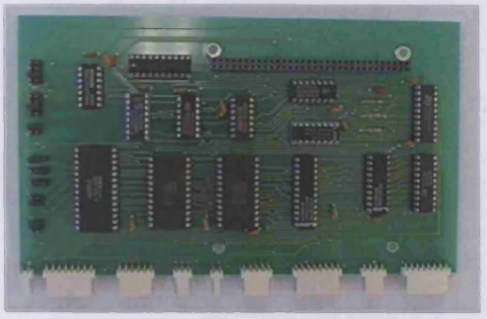


Figure 3-3. The Interface and Motor Control Board.

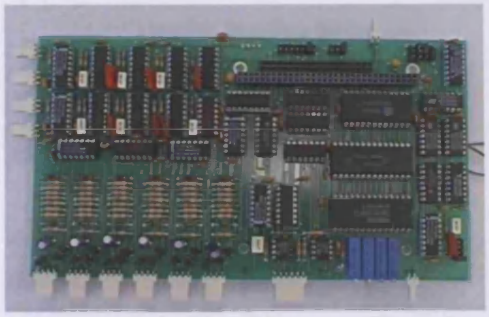


Figure 3-4. The Distance Board.

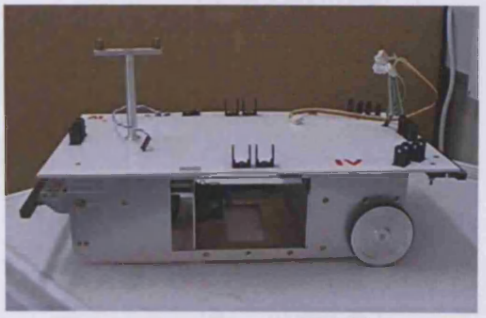


Figure 3-5. The BOM Robot Chassis.

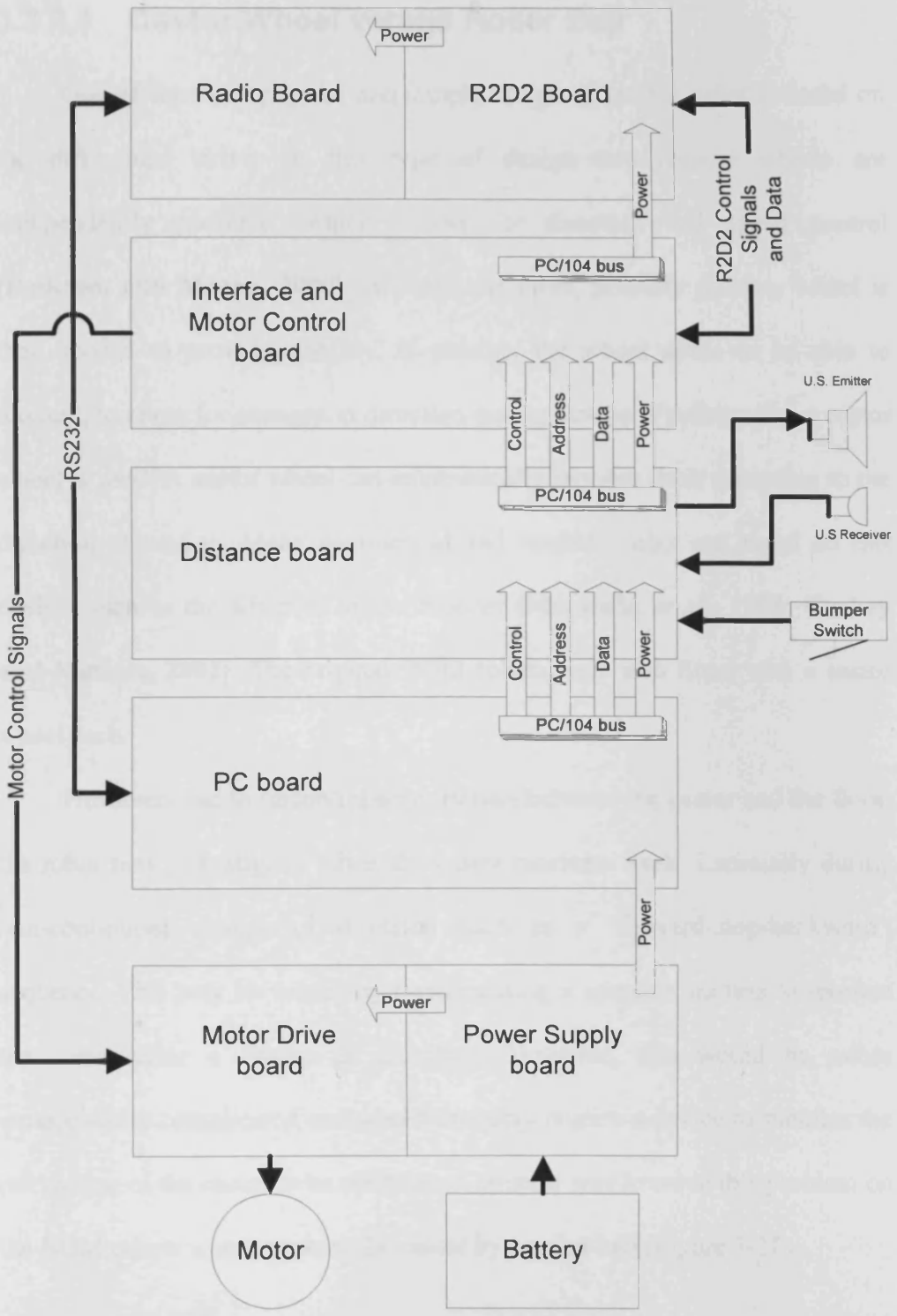


Figure 3-6. Hardware Structure of Original BOM Robots.

3.3.3.1 Castor Wheel versus Roller Ball

One of the most popular and simple design of mobile robot is based on the differential drive. In this type of design two coaxial wheels are independently powered, which allows for direction and speed control **[Balkcom and Mason, 2000]**. At least one more, possibly passive, wheel is then needed to provide stability. If passive, the wheel needs to be able to reorient, to allow for changes in direction during motion. Traditionally, a castor wheel is used. A castor wheel can automatically reorient itself according to the direction of motion. Many commercial and research robot are based on this design, such as the Khepera or the Pioneer **[Mondada *et al.*, 1993; Gerkey and Mataric, 2002]**. The original BOM robots were also fitted with a castor wheel each.

However, due to uncontrollable friction between the castor and the floor, the robot may jerk slightly when the castor reorients itself. Especially during non-continuous change of direction such as a 'forward-stop-backward', sequence. This may be solved by implementing a complex motion to reorient the castor after a change in direction. However, this would be rather unnecessarily complicated and would probably require a device to monitor the orientation of the castor to be efficient. A simpler way to solve this problem on the BOM robots was to replace the castor by a roller ball (Figure 3-11).

3.3.3.2 A New Operating System

The previous operating system running on the robots was MS-DOS (version 6.2). Despite being stable and easy to use when controlling hardware, and having a small footprint, MS-DOS had three major drawbacks. Firstly, it is not a multi-tasking operating system. Secondly, an increasing number of new hardware devices is not supported under MS-DOS. Thirdly, it is not an open source operating system and most of the applications distributed under it are in binary form. When it was chosen as the operating system for the robot in 1996, MS-DOS was the most suitable operating system available.

> *"The choice of the operating system (OS) was also based on compatibility considerations regarding other hardware components and their driver libraries. MS-DOS Version 6.2 was chosen because this fully documented and supported OS gave considerable advantages over not so popular OS, such as Mac."* **[Beutler, 1998 p.3]**

Because MS-DOS is not a multi-tasking operating system, only one application can run at a time. This means that the application developed to control the robot had to include everything required. This limitation did cause problems in the past. For instance, when trying to use a web-cam in the control program, previous developers had to write the code for image capture, exposure time, etc. Under a multi-tasking operating system, one can use a ready-made application for image capture, in parallel with its control

application. Furthermore, the program distributed under MS-DOS to capture images with this web-cam is in a binary form (precompiled). If the source code was available, the development of image capture functions for the control program would have been much easier. Finally, because most new hardware is not supported under MS-DOS, future development of the robot would be much more difficult. For instance if the robot were to be fitted with wireless network capability or a GPS device, the choice would be restricted because most new hardware is not supported under MS-DOS.

The problem caused by the limitation of MS-DOS highlighted the need for a more versatile operating system. This was an essential step in ensuring that the robots remain a versatile research tool and avoid becoming obsolete. Several operating systems were reviewed **[Corre, 2001]**.

Linux was chosen because it is a multi-task and multi-threads operating system. It is possible to develop a small footprint distribution that would match the hardware requirements. Linux is stable, well documented and widely used. A device driver is required to control peripherals but documentation on driver development is available. There is full, well documented networking support. Device drivers are generally available for most commercial hardware, such as the most popular wireless LAN cards and web-cams. In addition, if required in the future, a real time extension of the kernel is available to deal with time critical tasks.

As part of the development of the new operating system, a major concern was to ensure that interfacing with the motors and sensors of the robot is made easy. In order to enable simple and efficient ways to interface with them, from either a shell or any application, device drivers were developed for both the interface board and the distance board.

3.3.3.3 Wireless Communication

Another common problem in robotics is wireless communication. Efforts have been made throughout the evolution of robotics to free robots from umbilical cords. This was seen as an important step toward autonomous robots **[Warwick *et al.*, 1995]**. Advances in battery technology and electronics made it practical for robots to have onboard power supply. Most robots are now designed free of power supply cables. A further problem was the wireless communication medium. Such a system is often required for remote controlling, monitoring or inter-robot communication. The BOM robots were originally fitted with a radio board, allowing simple communication between the robots themselves, and/or a stand alone PC. This link was, however, very limited in speed, range and reliability.

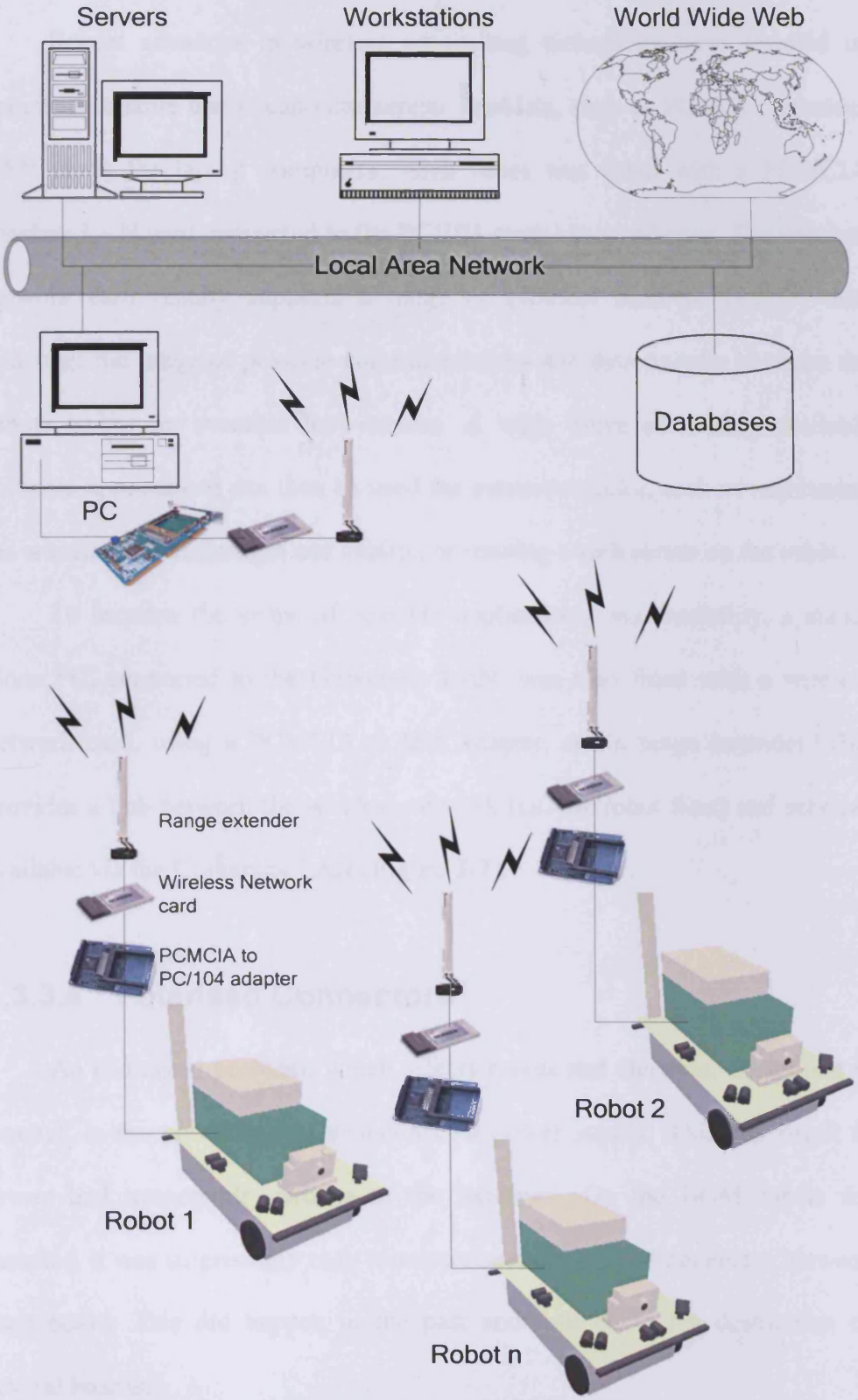


Figure 3-7. The Wireless Network Architecture.

Recent advances in wireless networking technology have resulted in compact, reliable and cheap commercial products, such as PCMCIA wireless LAN cards for laptop computers. Each robot was fitted with a PCMCIA wireless LAN card connected to the PC/104 stack via an adaptor. The wireless network card readily supports a range of protocol such as TCP/IP, thus widening the range of possible communications and data transfer between the robots and/or an eventual base station. A wide range of readily available software applications can then be used for numerous tasks, such as monitoring the wireless signal strength and quality, or running a web server on the robot.

To broaden the scope of possible applications and flexibility, a stand-alone PC, connected to the University LAN, was also fitted with a wireless network card, using a PCMCIA to ISA adapter, and a range extender. This provides a link between the wireless network (i.e. the robot fleet) and services available via the University LAN (Figure 3-7).

3.3.3.4 Polarised Connectors

An additional problem, which affects robots and electrical equipment in general, is the possibility to misconnect a power supply. This can result in severe and irreversible damage to the hardware. On the BOM robots for instance, it was surprisingly easy to misconnect the PC/104 connector between each board. This did happen in the past and resulted in the destruction of several boards.

This was easily solved by using non-symmetrical spacers between each board, (Figure 3-12) as recommended in the PC/104 standard **[Consortium, 2001]**, thus making it physically impossible to misconnect the boards in the stack. Another flaw in the original design was the connection between the power supply board (PSU board) and the PC/104 stack. Ideally the PSU should be fitted with a PC/104 connector and spacer and be part of the stack. This is not the case on the BOM robots. Originally, a wire was connected between the PSU board and the PC/104 connector at the top of the stack. This could, once again, be misconnected too easily. To solve the problem, a small, power distribution board was added to the stack. The board was designed so that it cannot be misconnected in the stack. The board was also fitted with a connector for a wire coming from the PSU board, and an auxiliary power output connector used to power the robot's onboard flash memory. Again, those connectors are designed to prevent misconnection. Finally, the link between the battery and the PSU board can be misconnected. This was not resolved, as the PSU board is designed to prevent damage in case of a misconnection. However, a battery designed in such a way that it cannot be inserted, or does not make contact, if it is inserted wrongly, would be a better solution.

Figure 3-8. The Fleet of Improved BOM Robots.

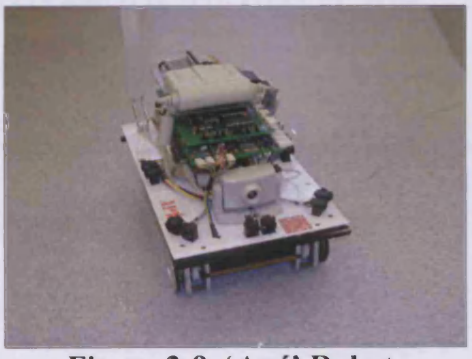


Figure 3-9. 'Apé' Robot.

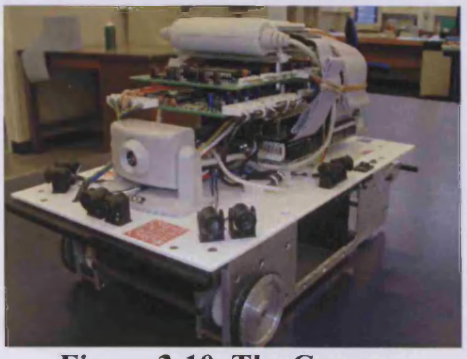


Figure 3-10. The Camera.

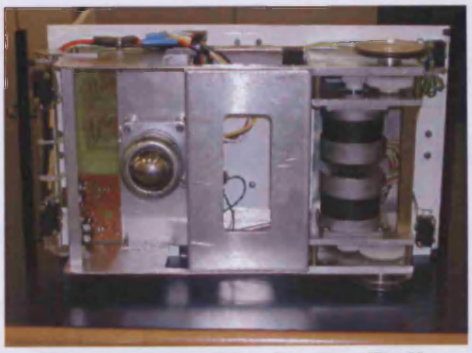


Figure 3-11. The Roller Ball.

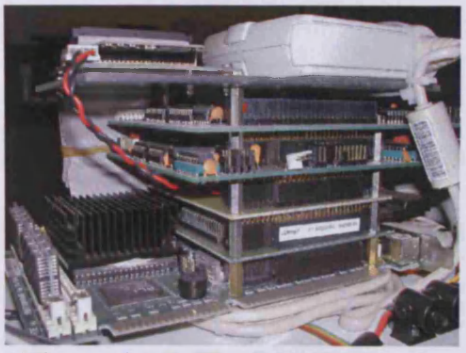


Figure 3-12. The PC/104 Stack.

3.4 Formal Concepts of the Architecture

3.4.1 Aims of the Architecture

It has been established that there is a need for development architectures that optimise research and implementation of multi-robot systems (MRS). The following objectives for such architectures were identified as necessary to provide versatile solutions:

- Scalability
- Easy embodiment
- Multi-user

The first objective is scalability. To ensure that the architecture can be used with different types of multi robot system and remain efficient it must be fully scalable. That is, the number of robots must not significantly affect the performance of the system. A simple action that may seem harmless when setting up one robot may prove a real chore for setting up a hundred robots.

The second objective is to optimise the process of implementing algorithms or control architectures within MRS, in other words, ease the embodiment process.

Finally, the architecture must remain efficient with a growing number of users developing or testing largely different algorithms or control architectures at virtually the same time.

3.4.2 Axioms

The Re-Embodiment architecture presented here is based on the notions of embodiment previously described. A system is embodied if a perturbatory channel exists between that system and its environment. Some of the concepts of the Re-Embodiment architecture are also based on the basic idea of dualism that a system consists of a soul and a body. This section formally postulates the axioms along which the Re-Embodiment architecture functions.

Axiom 1: There exists an environment.

One of the basic requirements of embodiment as defined by Quick *et al.* **[Quick and Dautenhahn, 1999; Quick *et al.*, 1999a; Quick *et al.*, 1999b]**, is the existence of an environment. In this research, whether this environment is physical or virtual, to avoid limiting the applicability of the architecture is not specified. The only prerequisite is that the environment is dynamic, in other words, it has at least one physical dimension: time. In a willingness to broaden application of the Re-Embodiment architecture, the boundary of the environment or what the environment consists of is not specified either.

Axiom 2: Among the entities in this environment, there exist bodies and souls. Both souls and bodies can be present and contained in this environment.

This environment may contain or consist of a variety of entities depending on its dimensions. Inspired by the fundamental ideas of Cartesian dualism, it is stipulated that among those entities, there exist bodies and souls. As Re-Embodiment makes use of bodies and souls, it is a prerequisite that both souls and bodies exist in the environment. It is also supposed that bodies and souls - by being present in the environment - can be identified and quantified along the dimensions of the environment. For instance, in the 'real world' environment, one should be able to say that at time t, the body B_n, of dimension $L_{B_n} \times W_{B_n} \times H_{B_n}$, was at location $T(B_n,t):\{x,y,z\}$ with the orientation $R(B_n,t):\begin{bmatrix} r_{11} & r_{21} & r_{31} \\ r_{12} & r_{22} & r_{32} \\ r_{13} & r_{23} & r_{33} \end{bmatrix}$. See Figure **3-13**.

Axiom 3: A soul can interact with a body to form a system.

Based on the existence of bodies and souls in the environment (as defined in axiom 2), it is specified that a system, in this environment, consists of the incarnation of a soul with a body. A system here has the same meaning than in the definition of embodiment by Quick *et al.* **[Quick and Dautenhahn, 1999; Quick *et al.*, 1999a; Quick *et al.*, 1999b]**. By incarnation, it is meant that for the soul and body to form such a system, the soul needs to be able to interact with the body and vice-versa. Note that the soul does not necessarily have to be physically contained in the body -along the dimension of the environment-. This again provides a larger scope of possible applications, by including the possibility of tele-embodiment. Tele-embodiment refers here to the interaction of a soul and a body each at different locations in the environment. See Figure **3-14**.

Axiom 4: The soul-body system can interact with its environment.

It has been established that a system consists of a soul and a body interacting. This, per se, is not sufficient for the system to be embodied. The soul-body system must also be able to interact with its environment. This is a prerequisite of the definition of embodiment advocated by Quick *et al.* **[Quick and Dautenhahn, 1999; Quick *et al.*, 1999a; Quick *et al.*, 1999b]**. See Figure **3-15**.

Axiom 5: A soul is not body specific.

This means that a soul should be able to incarnate in any body, thus providing maximum flexibility to the development system. See Figure **3-16**.

Axiom 6: A body is not soul specific.

This means that a body should be able to be incarnated by any soul. Again, this is done to provide maximum flexibility to the development system. See Figure **3-17**.

Axiom 7: A trigger will initiate the incarnation process. This trigger may come from the body, the soul or the environment.

The incarnation process corresponds with the time t in the environment when a soul and a body start to interact. This process may be initiated by a combination or either one of the following: A body, a soul or the environment. See Figure **3-18**.

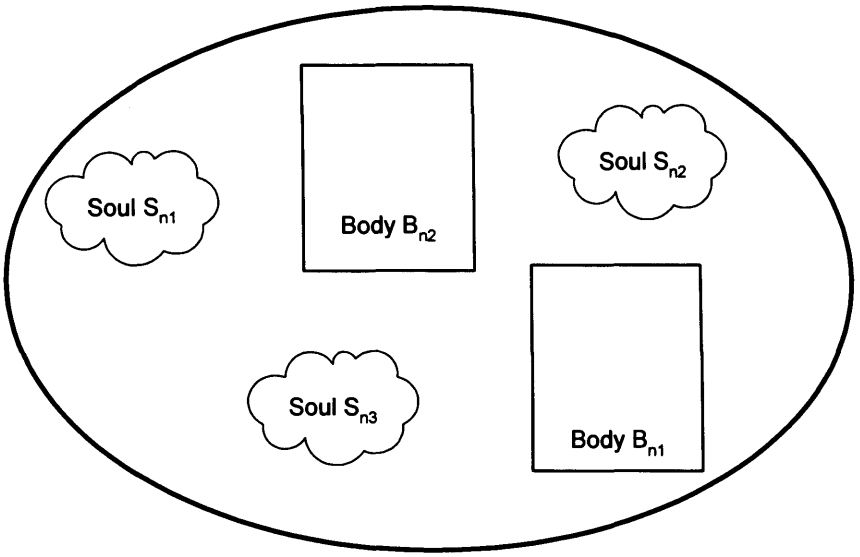


Figure 3-13. An Environment Containing Souls and Bodies.

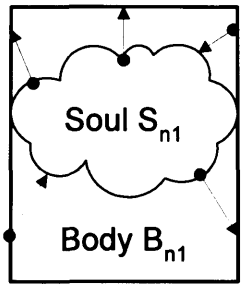


Figure 3-14. A Soul Interacting with a Body, Forming a System.

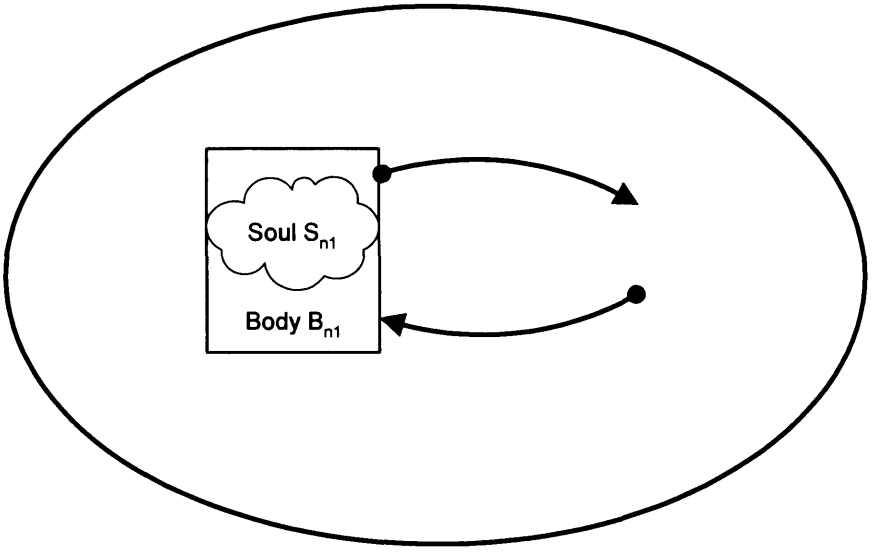


Figure 3-15. A Soul-Body System Can Interact with its Environment.

Axiom 8: A trigger will initiate the de-incarnation process. This trigger may come from the body, the soul or the environment.

The de-incarnation process corresponds with the time *t* in the environment when a soul and a body stop interacting. This process may be initiated by a combination or either one of the following: a body, a soul or the environment. See Figure **3-19**.

Axiom 9: A body may be linked with some information, body-specific or not. The active soul may access that information.

A body may also be linked with data and information. This data may be body specific or not. A soul incarnated with a body (i.e. an 'active' soul) may access that information. See Figure **3-20** below.

So far, the axioms (summarised in Table **3-1**, below) along which Re-Embodiment architectures are to be developed have been generally non restrictive. This was purposely done to broaden the scope of possible implementations and applications. However, as they stand, these axioms may be seen as too general for the implementation of development architectures for multi robot systems. For instance, they do not restrict the use of a single centralised soul to control several bodies.

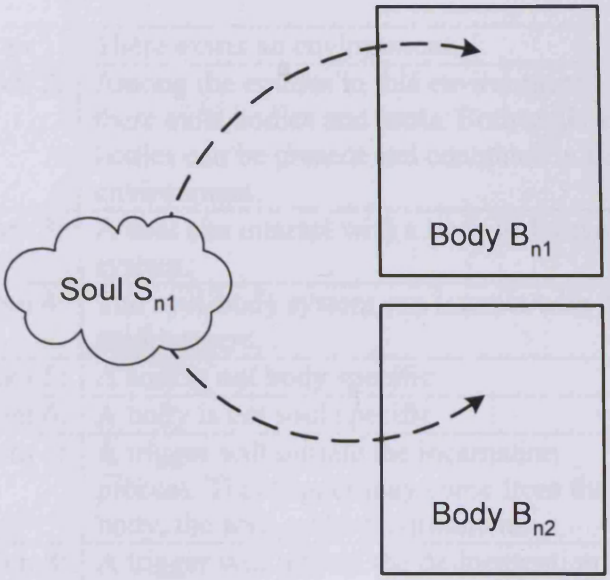


Figure 3-16. A Soul is not Body Specific.

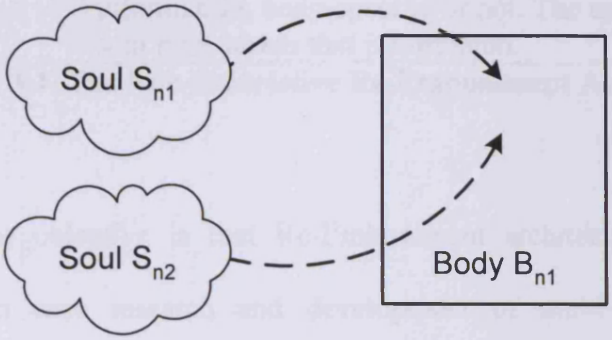


Figure 3-17. A Body is not Soul Specific.

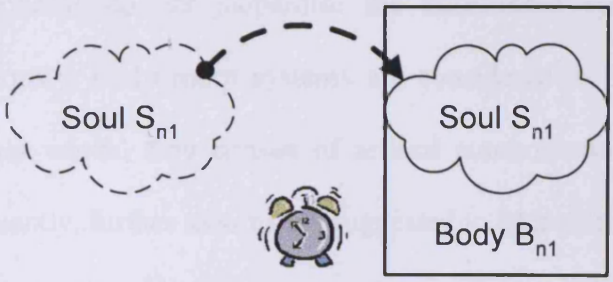


Figure 3-18. A Trigger Initiates the Incarnation Process.

Axiom 1:	There exists an environment.
Axiom 2:	Among the entities in this environment, there exist bodies and souls. Both souls and bodies can be present and contained in this environment.
Axiom 3:	A soul can interact with a body to form a system.
Axiom 4:	The soul-body system can interact with its environment.
Axiom 5:	A soul is not body specific.
Axiom 6:	A body is not soul specific.
Axiom 7:	A trigger will initiate the incarnation process. This trigger may come from the body, the soul or the environment.
Axiom 8:	A trigger will initiate the de-incarnation process. This trigger may come from the body, the soul or the environment.
Axiom 9:	A body may be linked with some information, body-specific or not. The active soul may access that information.

Table 3-1. The Non-Restrictive Re-Embodiment Axioms.

The prime objective is that Re-Embodiment architectures are to be implemented to ease research and development of multi-robot systems. Therefore, one must ensure that Re-Embodiment architectures developed along the proposed axioms do not jeopardise the multi-robot system itself. As discussed previously, multi-robot systems are considered to be decentralised systems. In other words, they consist of several autonomous subsystems, or robots. Consequently, further axioms are suggested to address this matter.

Axiom 10: A soul can only interact with a single body at a time.

This restriction ensures that a soul cannot control more than one body at a time and consequently that development architectures based on the Re-Embodiment axioms remain decentralised systems. See Figure **3-21**.

Axiom 11: A body can only interact with a single active soul at a time.

This restriction ensures that a body is only controlled by a single soul at a time. Therefore, development architectures based on the Re-Embodiment axioms are not plagued with resource sharing problems. Note that, for ease of implementation, a body may contain other souls but only one can form part of the soul-body system. See Figure **3-22**.

Axiom 10:	A soul can only interact with a single body at a time.
Axiom 11:	A body can only interact with a single active soul at a time.

Table 3-2. The Restrictive Re-Embodiment Axioms.

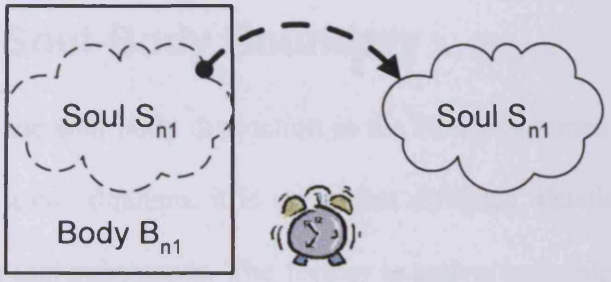


Figure 3-19. A Trigger Initiates the De-Incarnation Process.

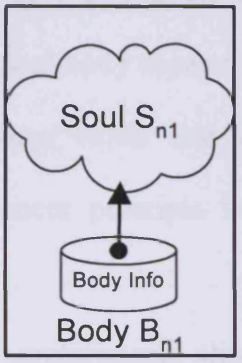


Figure 3-20. A Body May Contain Data that the Active Soul May Access.

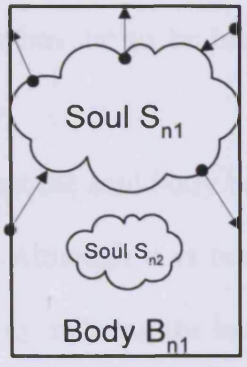


Figure 3-21. A Body Can Only Interact with One Soul. A Soul Can Only Interact with One Body.

3.4.3 The Soul-Body Boundary

Although the soul body distinction in the Re-Embodiment architecture is inspired by Cartesian dualism, it is somewhat different. Dualists see soul and body as two distinct substances. The former is active, invisible and intangible, while the latter is inactive, visible and tangible. This forms the basis of one of the most pertinent criticisms of dualism, the category mistake. In the Re-Embodiment architecture, the soul-body separation is set arbitrarily. Both soul and body are part of the same world and are not different substances. Consequently the Re-Embodiment principle is unaffected by the category mistake.

In the Re-Embodiment architecture, this soul-body boundary is set according to the system itself and its implementation. As long as the axioms, along which development systems are to be implemented, are honoured, this boundary can be set arbitrarily.

It is important to note that the soul-body boundary is independent of any software-hardware boundary. Although it is tempting to imagine the soul as being the software and the body as being the hardware, this would restrict the applicability of the architecture. In most systems, both body and soul will be made of a mixture of software and hardware. For instance, a body may consist of a chassis, motors, sensors, and controller. There also may be some resident software, it may for example hold some information specific to the body and required by the implementation, such as a network address. The soul may consist of control programs and part of the operating system. It may also

include some hardware. One could implement a system where the soul is a data storage medium, such as a CD or a data cartridge, loaded with software. This storage medium can then be inserted in the body, during the incarnation process. Soul and body will then interact to form the complete system.

3.4.4 Agent Systems versus Re-Embodiment

Another distinction, which may need to be clarified, is the one between so-called Software Agent Systems (SAS) and the Re-Embodiment architecture. The term agent system is used to refer to several different things. It may refer to a fleet of robots co-operating in the physical world, simulated ant colonies, or small software programs foraging in the Internet in a decentralised manner. These software agent systems may exhibit some similarity to the Re-Embodiment architecture, in that a software agent system may be used to control a fleet of robots **[Beutler, 1998]**. Both software agents and souls in a Re-Embodiment architecture may be able to 'freeze' and move to another machine or body to carry on running. There are, however, major differences. In software agent systems, several agents may be running at the same time on the same machine and may need to compete for resources. This is not the case in Re-Embodiment architecture. According to axioms 10 and 11, a soul can only interact with a single body at a time and vice versa.

It is possible to implement a Re-Embodiment system where a fleet of robots is controlled by travelling software agents. In this very particular case, the software agents are a subset of the soul. The souls to be incarnated on the robot will contain the necessary components to run the software agent system,

such as an agent server and several software agents. Once the incarnation process is completed, the software agent system may start running.

3.5 Re-Embodiment Applied

The axioms according to which Re-Embodiment development architectures are to be implemented have been established. However, any implementation issues have been abstracted so far. Now that the formal axioms have been defined, one can envisage possible applications of the Re-Embodiment architecture. This section presents several possible applications in which a Re-Embodiment architecture would ease research and development of multi robot systems. Note that this list of possible applications does not attempt to be exhaustive. The distinct advantages of such architecture are also highlighted in each case.

3.5.1 Easy Re-Embodiment

The first, and obvious, advantage of Re-Embodiment architectures is, as the name suggest, Re-Embodiment. As defined previously a system consists of a body and a soul. While developing a robot or a multi-robot system, a large amount of time is spend on debugging and testing. This can be on either the soul or the body. One could be testing a new image capture routine, part of the soul, or a new control board for sensors, part of the body. In Re-Embodiment architectures, the process by which a soul incarnates a body to form a system is simplified; therefore, the development process can be optimised. Users can

work on different parts of the system and easily bring the soul and body back together to have a system in a running state. This process can then be repeated throughout the development and test period. See Figure **3-22**.

3.5.2 Multi-Embodiment

Another distinct advantage of the proposed architecture is multi-embodiment. Researchers working with a fleet of robots can readily incarnate several instances of the same soul into several distinct bodies of the fleet. This is particularly advantageous with growing numbers of robots. The time required to bring each robot system to a running state becomes increasingly critical as the number of robots increases. The development process can be optimised by making such a task easier and faster. Development architectures based on the Re-Embodiment principle will provide such capabilities. See Figure **3-23**.

3.5.3 Cloning

Another possible advantage of Re-Embodiment architecture is cloning. As part of a multi robot system, one can easily implement a cloning process. For instance, a soul-body system could duplicate itself. Several new instances of the soul can be embodied in new bodies. It is assumed that the new bodies used for the cloning are available for incarnation. See Figure **3-24**.

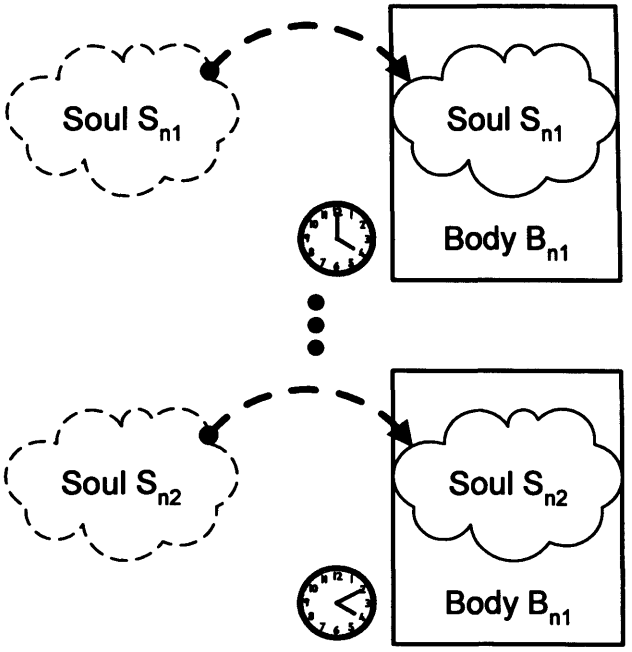


Figure 3-22. Easy Re-Embodiment.

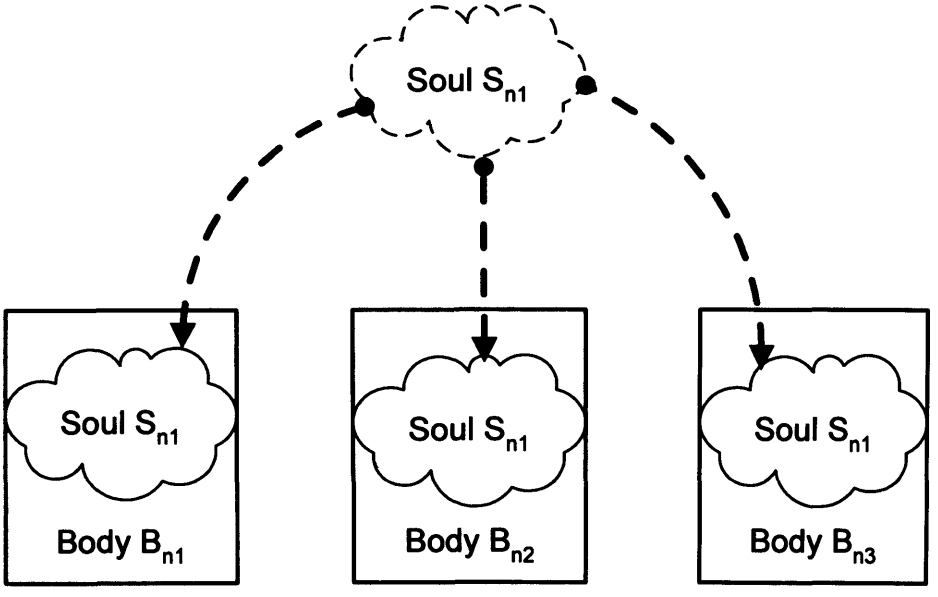


Figure 3-23. Multi-Embodiment.

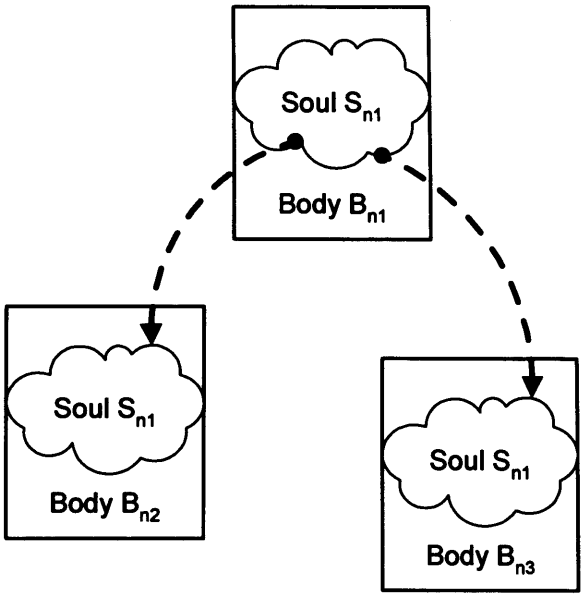


Figure 3-24. Cloning.

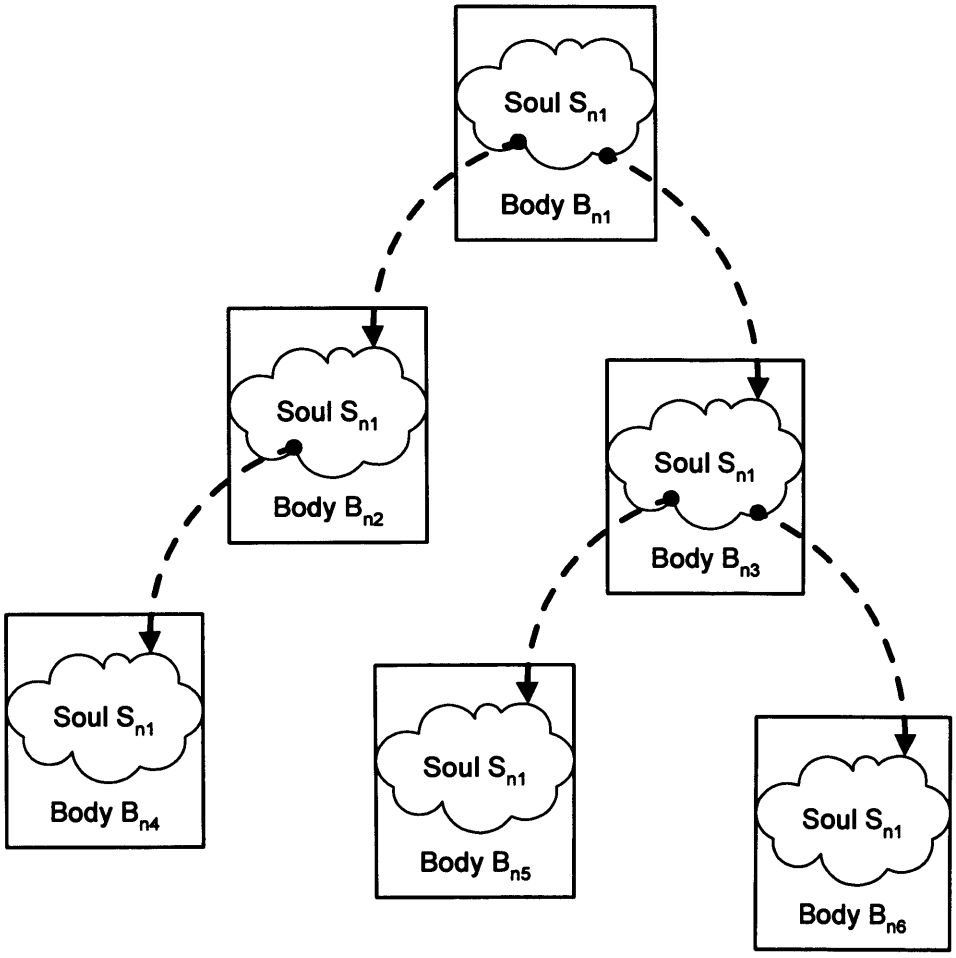


Figure 3-25. Deployment.

3.5.4 Deployment

A further advantage of the system is deployment. A soul can be easily deployed across a fleet of bodies. It is particularly advantageous in cases where there are restrictions in the communication range. If a soul cannot be incarnated in all the bodies from the start, it can be firstly incarnated in the bodies that are within range. Then, those bodies can clone instances of their soul into further bodies within their range. See Figure **3-25**.

3.5.5 Crossover

An additional possible use is crossover mutation. One can easily implement a society where at some stage two souls crossover to form a third soul. This third soul can then be easily incarnated in available bodies. This would be particularly advantageous in system where co-evolution is to be implemented with robots in the physical world. See Figure **3-26**.

3.5.6 Self-Mutation

A further possible use is self-mutation. One can easily implement a society where at some stage souls in robots systems mutate and re-incarnate in their own body. This would be particularly useful in cases where one wants to implement an evolutionary system using real robots. See Figure **3-27**.

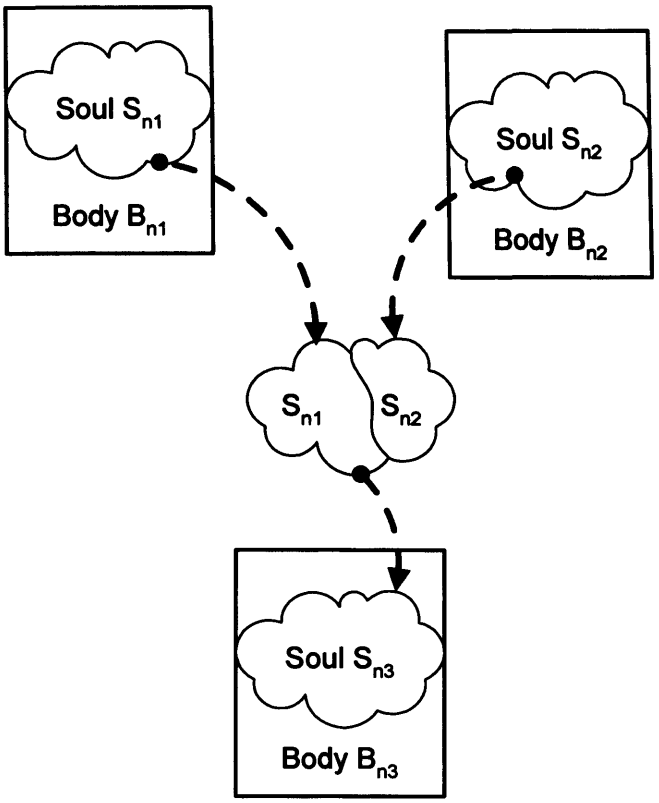


Figure 3-26. Crossover.

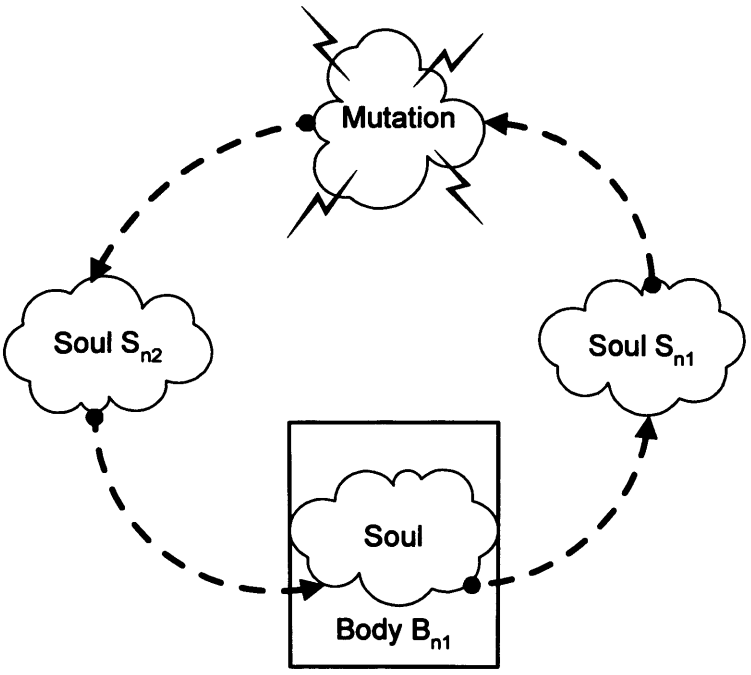


Figure 3-27. Self-Mutation.

3.5.7 Multi-User Scenario

The main advantage of Re-Embodiment architectures is their versatility with multiple users. Development exercises may require several users to test largely different things at virtually the same time. Several projects may be running in parallel with only one available platform for testing purposes. Here, a small scenario highlights the adaptability of a development architecture, based on the Re-Embodiment principles.

Imagine a research laboratory where three researchers are developing essentially different control algorithms. The laboratory is equipped with four mobile robots for testing and development purposes. The researchers have to share these resources to test their theories (See Figure **3-28**).

Researcher 1 is developing a control algorithm to track an object with the camera. He prepares a soul, containing all required drivers to interface with the robot hardware and the camera. It also includes a program where his algorithm is coded. At 4pm, he triggers the incarnation of two instances of his soul in robots 1 and 2 and commences testing.

Researcher 2 is developing control algorithms for robots to follow each other. She prepares two souls, one with her algorithm coded in a program, one with a program to make a robot randomly wander. At 4.10pm, she triggers the incarnation of her two souls in the two available robots, respectively robots 3 and 4. Soon after, robot 4 starts wandering in the lab, while robot 3 follows it.

At 4.20pm, researcher 1 ceases his test session and starts working on his algorithm again to correct minor unexpected problems he was able to uncover with his tests.

At 4.25pm, researcher 2 terminates her test session happy with the results.

Researcher 3 is developing control algorithms for moving robots in a formation. He has previously prepared and tested a soul, containing his algorithms. At 4.30pm, he triggers the incarnation of instances of his soul in all available robots for a demonstration to visiting researchers.

At 4.45pm the demonstration is finished and the visiting researchers are impressed.

At 4.50pm, researcher 2 triggers another incarnation to test a new algorithm she is developing.

This scenario aimed to demonstrate that the use of a single development platform, based on Re-embodiment, is easily optimized to a point where several researchers can work in parallel on different aspects of MRS and robotics research.

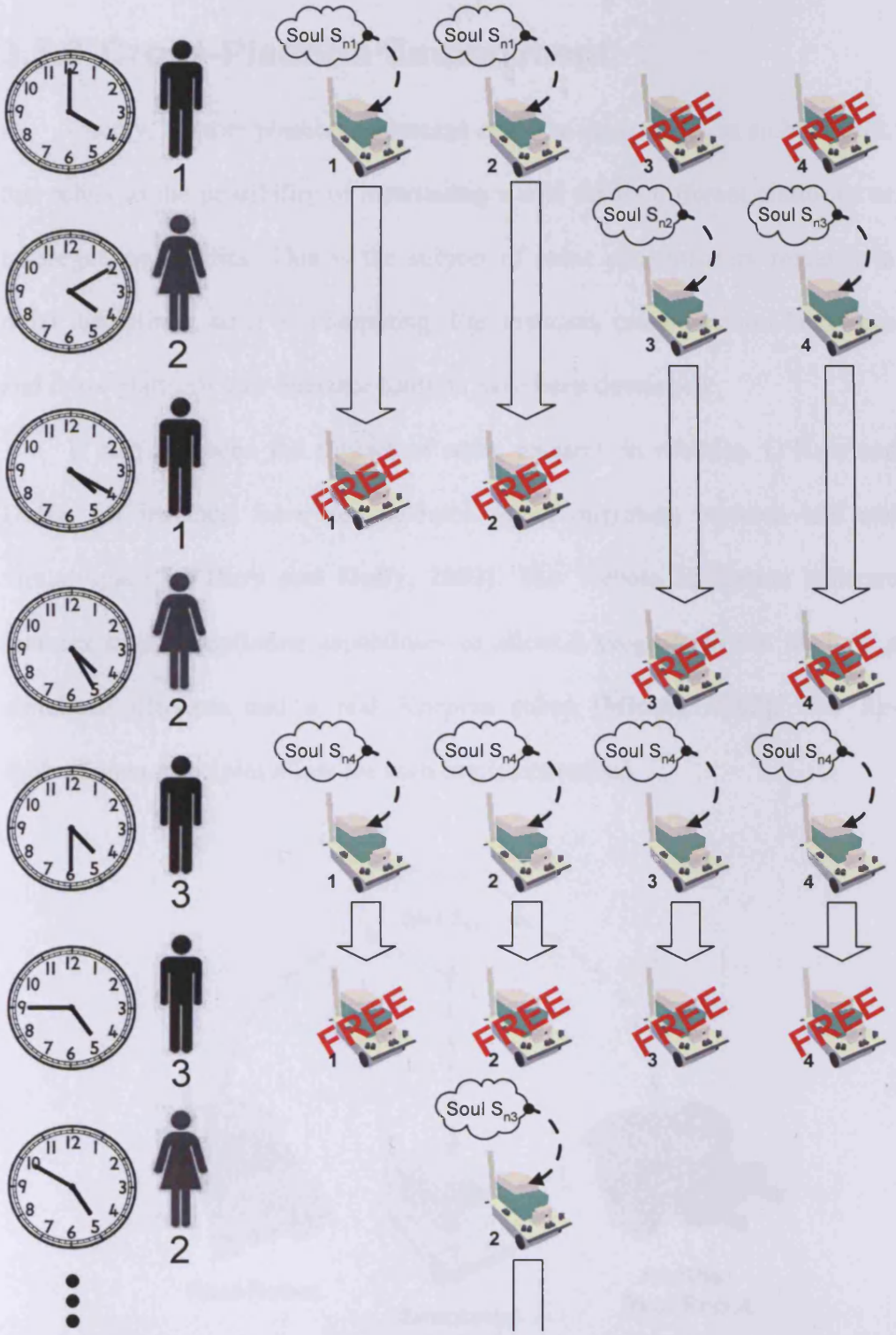


Figure 3-28. Multi User Scenario.

3.5.8 Cross-Platform Embodiment

Finally, another possible advantage could be cross-platform embodiment. this refers to the possibility of incarnating a soul across different platforms or heterogeneous bodies. This is the subject of some contemporary research in other disciplines, such as computing. For instance, cross-platform languages and cross-platform user interface toolkits have been developed.

It also has been the subject of some research in robotics. O'Hare and Duffy, for instance, have demonstrated agent migration between real and virtual space **[O'Hare and Duffy, 2002]**. The Webots simulation software features cross-compilation capabilities to allow a program to run on both a simulated platform and a real Khepera robot **[Michel, 2003]**. The Re-Embodiment principles allow for such implementations.

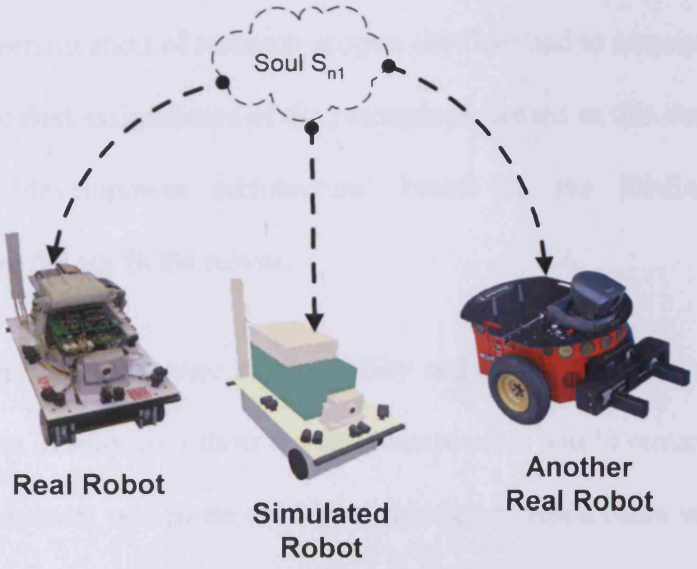


Figure 3-29. Cross Platform Embodiment.

3.6 Implementing Re-Embodiment Architectures

In this section, two development architectures based on the Re-Embodiment principles are described. The first was implemented on the Bunch of Mobile robots (BOM robots), a fleet of six PC based robots. The second was implemented on small robots code-named SheepBots. The SheepBots are a large fleet of microcontroller based robots.

3.6.1 Re-Embodiment on the BOM Robots

The BOM Robots were developed at Cardiff University **[Beutler, 1998]**. Six were built. This small fleet has been used by the robotics group over the past few years as the hardware platform to implement, test, and validate control algorithms **[Pham and Parhi, 2003]**. With a growing number of researchers and a wider diversification of research scopes, the fleet had to remain a flexible tool. One of the first assignments of the research presented in this thesis was to implement a 'development architecture' based on the Re-Embodiment principles, using the six BOM robots.

The main objectives were expandability and flexibility. The system was to consist at first of only six robots but future expansion was to remain possible and easy. The system was to be used by a number of researchers working on largely different things, hence the need for flexibility.

The first step was to devise the development architecture structure based on the Re-Embodiment concepts. Then a new custom-made operating system implementing the Re-Embodiment principles was developed for the robot. Device drivers were developed to control the robots' specific hardware under this operating system **[Corre, 2001]**. Finally, the system reliability and performance were assessed through both testing and continuous use by the researchers of the robotics group.

3.6.1.1 System Design

The robots are to be used in the real world. According to axiom 1, the environment for the robots will be the physical world.

The robots are PC-based; the body of each robot will consist of all the hardware, and some resident software. Because the robot may be required for a number of tasks, the software part of the body will be kept minimal. It will consist of the operating system kernel and some body specific information.

The soul will be purely software. It will consist of the entire File System (FS) needed by the operating system and the robot to run. This allows for maximum flexibility. For instance, if one does not need to use the robot's camera for a particular experiment, then one may omit the corresponding device driver and sample applications from the file system. The file system will include one or more programs where the algorithms to be tested are coded. It may also include programs and utilities, such as a shell, a telnet server for

remotely logging on to the robot, or a web server. The soul and body have been defined in accordance with axiom 2.

The file system will also include some device drivers for the motors and sensors, allowing the soul to be able to interact with the body. Applications or user programs will also be able to interact with the body via these device drivers. A file containing the body-specific information will also be made available. Consequently, the soul will be able to interact with the body, hence fulfilling axiom 3.

The soul-body system consists of a Central Processing Unit (CPU), actuators (i.e. the motors, sensors,) and control programs. The system, once running, will be able to interact with its environment. In other words, the robot will be able to move, push objects, and dynamically affect other systems in the environment. It will also be able to perceive this environment through various sensors, such as bumpers, ultrasound range sensors, and vision sensors. Axiom 4 is thus also fulfilled.

The system is designed so that souls remain non-body-specific. For this to be possible, the required information specific to a body is held on that body in accordance with axiom 9. This information, held in a file, is part of the software side of the body. For this implementation, the only body specific information required is the Internet Protocol (IP) address of the body on the wireless network, together with its hostname. Axiom 5 is therefore met.

The system is also designed so that a body is not soul-specific. In other words, each body can accommodate any soul, hence fulfilling axiom 6. There are, of course, a few provisos. One can assume that the souls are developed to run on those bodies. A researcher developing a soul containing inadequate device drivers or no program cannot expect the system to be embodied after the incarnation process.

According to axiom 7, a trigger initiates the incarnation process. To ease the implementation, a simple trigger was chosen. Each time a body's operating system reboots, it will look for a soul destined to be incarnated on that body. Souls, being purely software in this application, have to be held in a storage medium. When incarnated in a body they will be held in the memory of that body. When not incarnated, a computer, part of the wireless network, is made available for them to reside until an incarnation is triggered. This computer is referred to as the boot server. This computer is also connected to the University LAN. Users can develop their souls from any workstation on the University network or any workstation in the world connected to the Internet, provided there is authorised access to the University LAN. Souls, once ready, can then be transferred to that boot server.

This boot server is equipped with a specific directory structure. There exists a specific directory for each robot. A soul residing in one of those directories is considered to be waiting incarnation into the corresponding body. For instance, soul S_{n1} may be stored in the folder corresponding to body B_{n1}. When body B_{n1} is powered up, its operating system will download soul S_{n1} and

use it for the incarnation process. There also exist folders for groups of robots, and a folder for the entire fleet. For example, one may want instances of the same soul to be incarnated in the entire fleet. The only step required for this is to place this soul in the directory corresponding to the entire fleet and then power up, or reboot, every robot in the fleet.

A default soul is also stored and compressed in each robot as part of the body's software side. If there are no souls available on the boot server during the incarnation or if the available souls are invalid, this default soul is used for the incarnation. This allows for debugging of the incarnation process. After a failed incarnation from the boot server, a soul-body system is still created, thus allowing the user to investigate eventual problems. The default soul can also be used to quickly demonstrate the capabilities of the system, without having to prepare a soul and upload it on the boot server.

Axiom 8 stipulates that a trigger will initiate the de-incarnation process. This will be achieved by a reboot or a power down of the body's operating system. Before the de-incarnation, the soul may be saved and stored back on the boot server or re-incarnated in another robot. Otherwise, the soul is lost.

Axiom 1:	There exists an environment.	The real physical world.
Axiom 2:	Among the entities in this environment, there exist bodies and souls. Both souls and bodies can be present and contained in this environment.	A body: the entire robot hardware i.e. the chassis, motors, electronics, etc. The operating system's kernel. A file containing some body specific information. The linuxrc script. A compressed default file system image. A soul: Purely software, consists of a file system image containing required programs, servers, drivers, and applications.
Axiom 3:	A soul can interact with a body to form a system	Device drivers have been developed for applications and programs to interact with the robot's specific hardware. Device drivers are readily available for the robot's commercial hardware.
Axiom 4:	The soul-body system can interact with its environment.	The soul-body system is equipped with sensors and actuators allowing interaction with the environment.
Axiom 5:	A soul is not body specific.	A soul is free from any body-specific information, hence can be incarnated in any available body.
Axiom 6:	A body is not soul specific	A body is free to be incarnated by any consistent soul.
Axiom 7:	A trigger will initiate the incarnation process. This trigger may come from the body, the soul or the environment.	The incarnation process takes place during the boot up sequence of the operating system. A power up or reboot triggers the incarnation.
Axiom 8:	A trigger will initiate the de-incarnation process. This trigger may come from the body, the soul or the environment.	The de-incarnation process occurs when the system shutdowns or reboots.
Axiom 9:	A body may be linked with some information, body-specific or not. The active soul may access that information.	A file, part of the body's software, contains all required body specific information.

Table 3-3. The BOM Robot Implementation According to the Re-Embodiment Axioms.

3.6.1.2 Incarnation Process

This section details the incarnation process as it was implemented on the BOM robots. The incarnation process occurs as a part of the boot up sequence. Linux kernels have been designed to support boot loading via the use of the linuxrc script. This was originally developed as a workaround to enable the use of boot devices not directly supported by the kernel. That capability to implement the incarnation process was taken advantage of. This process is also summarised in Figure **3-31**.

1st stage: The robot is powered up

When a computer is first powered up, the BIOS starts the system. Then a program called the bootstrap loader, located in ROM BIOS, looks for a boot sector. A boot sector is the first sector of a disk and has a small program that can load an operating system. When the robot is powered up the BIOS checks the memory and all other devices. Then the BIOS executes the Linux Loader (LILO) located on the first sector of the flash disk.

2nd stage: The Linux Loader (LILO)

When the computer loads a boot sector on a normal Linux system, what it loads is actually a part of LILO, called the 'first stage boot loader'. This is a tiny program which loads and runs the 'second stage boot loader'. The second stage loader gives a prompt (if it was installed that way) and loads the chosen operating system.

On the robot, the Linux Loader extracts the kernel from the flash disk and decompresses it into the RAM. Then the Linux kernel is executed.

3rd stage: The Linux kernel

The kernel makes the hardware do what the programs want, fairly and efficiently. The processor can only execute one instruction at a time, but Linux systems appear to be running several processes simultaneously. The kernel achieves this by quickly switching from task to task. It makes the best use of the processor by keeping track of which processes are ready to go, and which processes are waiting. This kernel task is called scheduling. If a program is not doing anything, then it does not need to be in RAM. Even a program that is doing something may have parts that are not doing anything. The address space of each process is divided into pages. The kernel keeps track of which pages of which processes are being used the most. The pages that are not used so much can be moved out to the swap partition. When they are needed again, another unused page can be paged out to make way for them. This is virtual memory management. The kernel contains large amount of specific code to interact with diverse kinds of hardware and presents it in a uniform way to the application programs. The kernel also manages filesystems, interprocess communication, networking, etc.

On the robot, the Linux kernel initialises all devices and then extracts a default file system from the flash disk into the RAM. This default file system contains:

- File transfer applications (ftp, tftp, or nfs)
- The required device driver for the wireless network card
- A shell or command interpreter (bash or ash)
- A script (linuxrc)

Once the default file system is extracted into RAM, it is mounted as a default root file system and the linuxrc script is executed.

4th stage: The linuxrc script

The linuxrc script uses the bash command interpreter or shell to execute a sequence of task. First, the wireless network card is initialised. Then, the script checks if the boot server can be accessed and if there is a valid file system image available for download.

If there is a valid image, it is downloaded over the wireless network. It is then decompressed into RAM and mounted as the new root file system. Log files and the file containing the information specific to the body are transferred from the old root file system to the new one. The old root file system is unmounted. The boot process carries on running with the new file system.

If there is no valid image on the server, or if the server cannot be accessed, the script notifies the user by logging an error message. The boot process continues with the default file system.

This sequence is summarised in Figure **3-30** below. The full source code and flowchart are also available in Appendices B and C.

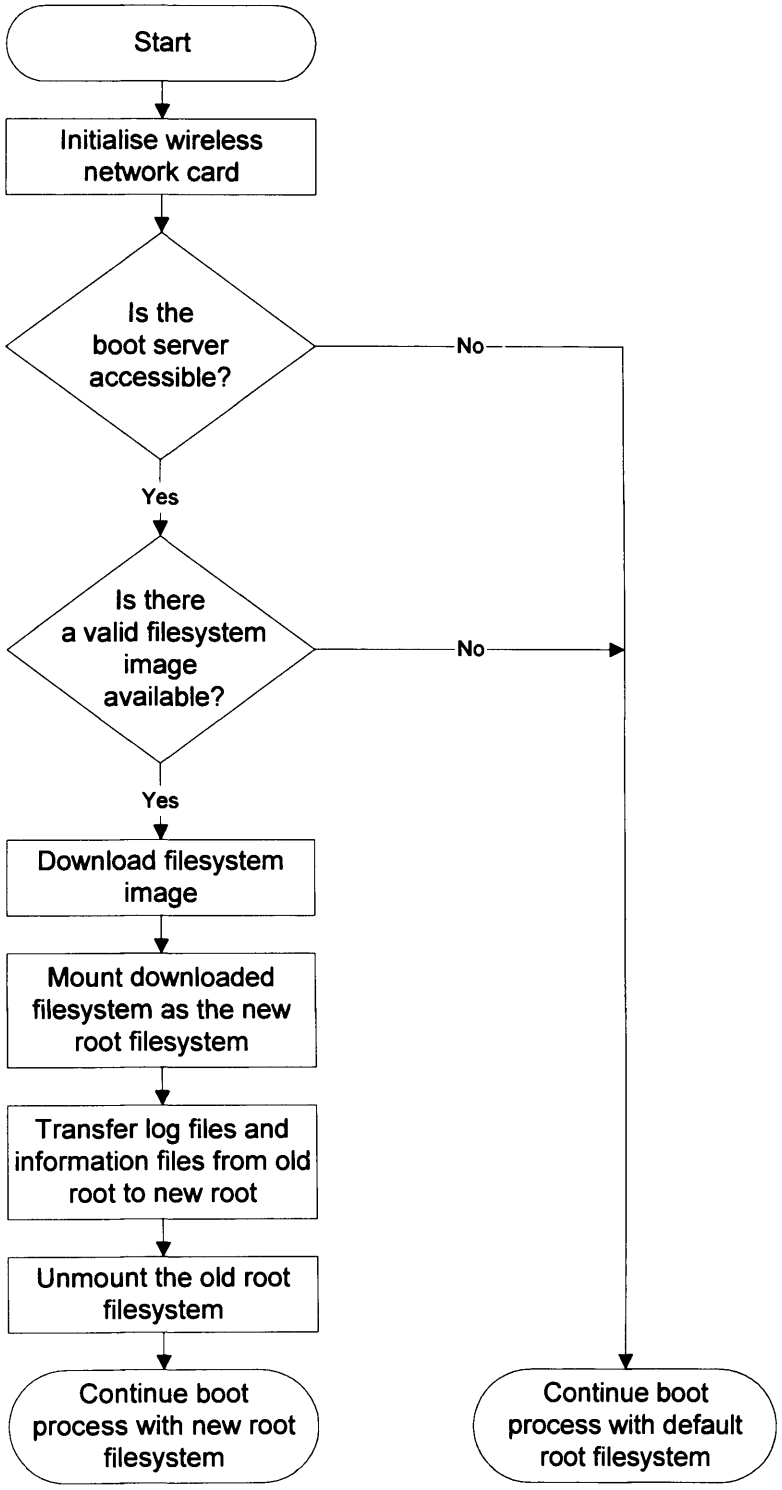


Figure 3-30. Flowchart Summarising the linuxrc Script

5th stage: The init program and other applications.

Only the "System V" style of initialisation used by most Linux systems will be described here. There are alternatives. In fact, one can put any program in '`/sbin/init`', and the kernel will run it.

Init's function is to ensure that that everything is running the way it should be. It checks that the filesystems are ok and mounts them. It starts up 'daemons' to log system messages, do networking, serve web pages, listen to the mouse, and so on. It also starts the '`getty`' processes that put the login prompts on your virtual terminals.

'Init' reads the file '`/etc/inittab`', which tells it what to do. Typically, the first thing it is told to do is to run an initialisation script. The program that executes (or interprets) this script is `bash`, the same program that gives a command prompt. In most Linux systems, the initialisation script is '`/etc/rc.d/rc.sysinit`'. This is where the file systems are checked and mounted, the clock set, swap space enabled, hostname set, etc.

On the robot, the 'init' program from the current root file system, (either the default root file system or a downloaded root file system), is executed as well as all other required programs, as defined in the '`/etc/inittab`' file.

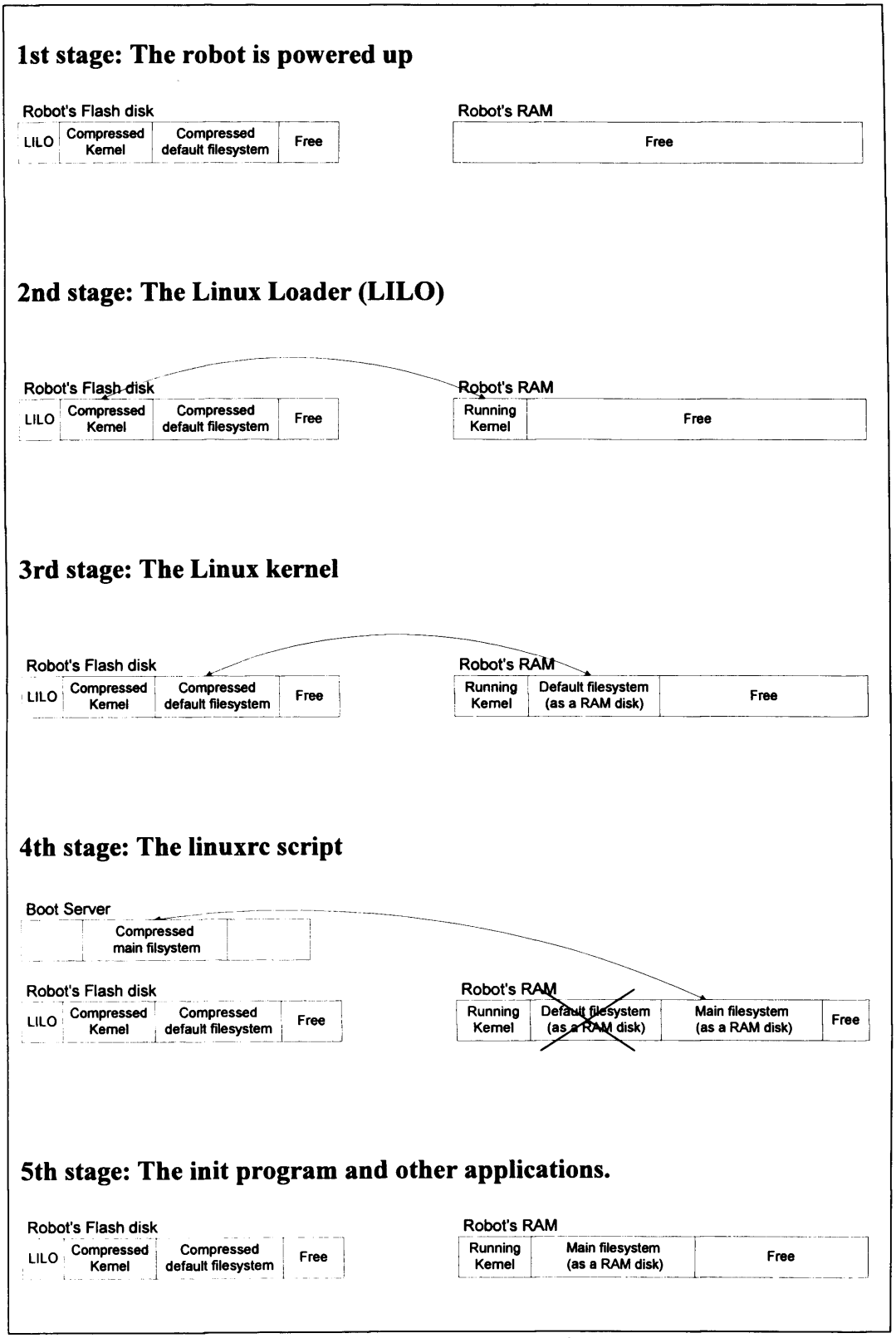


Figure 3-31. The Five Stages of the Incarnation Process on the BOM Robots.

3.6.1.3 Performance Assessment

The performance of the system was first assessed through a series of tests. The system reliability was also tested through constant use by researchers in the robotics group over the last couple of years **[Pham and Parhi, 2003]**.

The aim of the tests was to assess the performance of the system when downloading file system images. This was done by timing the boot sequence of each robot. It is important to note that the timing was done manually. The nature of the boot sequence does not allow software timing to be implemented. The time was measured from the moment were the robot is switched on and until the boot process is fully completed. Because the robots are not fitted with screens or other output devices, a double beep marks the end of that boot sequence. It is also important to note that the robots are not all fitted with the same PC board. One has an Intel 486 clocked at 33MHz and the other five have an Intel Pentium I clocked at 66Mhz. The full robot specifications are available in Appendix A.

The first series of tests was designed to time the boot sequence when no file system image is available on the boot server. Those experiments showed that the average boot time on the 486 is about 57 seconds, against 53 seconds on the Pentium. The full results from those experiments are available in Appendix D.

Then a second series of tests was run to time the boot sequence when a four megabytes image is available. Finally, a third series of tests was run to time the boot sequence when an eight megabytes image is available. Those experiments showed that the average boot time with a four megabytes image on the 486 is 90 seconds, against 76 seconds for the Pentium. With an eight megabytes image, average times are 102 seconds and 85 seconds respectively.

Finally, a series of tests was run where all robots simultaneously boot up. Those tests were firstly run with no image to download, then with a four megabytes image to download each and finally with an eight megabytes image to download each. The aim of these tests was to assess how the bandwidth of the wireless network card affects the boot process. As expected, the results are unchanged when there are no images to download. When there is a four megabytes image to download, the boot up time increases to about 103 seconds for the 486 and 89 seconds for the Pentium. With an eight megabytes image, the boot up time also increases to 138 seconds for the 486 and 125 seconds for the Pentium. This increase in the download time is due to the limit of the wireless card bandwidth. However, it does not dramatically affect the usability of the system. All the results are summarised in Table **3-4** below.

Average Boot-up Time				
	486 DX2 33MHz		Pentium I 66MHz	
	Individual (Seconds)	Simultaneous (Seconds)	Individual (Seconds)	Simultaneous (Seconds)
0MB	57.31	57.41	52.77	52.74
4MB	90.04	103.12	76.28	89.03
8MB	101.99	137.80	85.21	125.15

Table 3-4. Average Boot-up Times.

3.6.2 Re-Embodiment on the SheepBots.

Researchers at Cardiff University have been working on decentralised control systems relying on very large numbers of agents. The development and test platform based on the BOM robots and the Re-Embodiment principle has proved effective and flexible. However, the fleet only consists of six robots. Although the development architecture allows for expansions, the actual hardware design of the robot is cause for concern. The BOM robots are relatively large. To run an experiment involving a hundred robots would need a vast environment. The BOM robots, being PC based have good computational power but are relatively expensive. Generally, control systems relying on a very large number of agents may cope with reduced computational power for each individual. They rely instead on the sheer number of individuals in the system.

Therefore, there is a need for another hardware platform better suited to experiments involving large numbers of robots with relatively limited individual computational power. The main design requirements for the new fleet are:

- Very low cost per individual robot, thus allowing for the production of a large fleet with a reduced budget.
- Very small size to ensure that a fleet consisting of up to a hundred robots can be physically used in a realistically sized laboratory.
- Expandability. Again the platform must be fully extendable.

- Flexibility, again, is also a prime requirement.

Several prototypes were developed. A fleet of fifteen robots code-named SheepBots was built based on the first prototype. This fleet was used as part of a display at the BBC Tomorrow's World Road Show held in Cardiff in 2002 **[MEC, 2002]**. Each robot consists of a small chassis and a sheep-like cover. Two small motors and gearboxes power the robots. The control electronics are based on a 16F873 PIC micro-controller. Simple infrared sensors interfaced with the PIC allow for obstacle detection. However, this small fleet was built for a one-off display and remained inflexible.

The objective of the second prototype was to implement the Re-Embodiment principle to provide a more flexible hardware platform.

Figure 3-32. A SheepBot at the BBC Road Show.

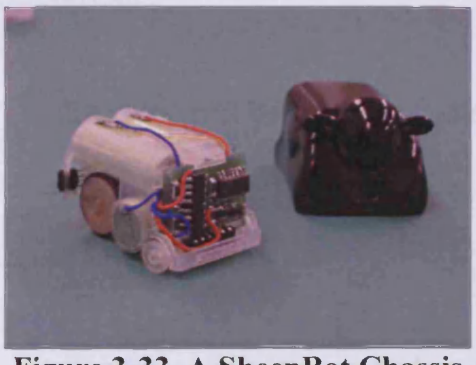


Figure 3-33. A SheepBot Chassis and Cover.

Figure 3-34. The SheepBots Flock.

Figure 3-35. SheepBots Chassis Assembled.

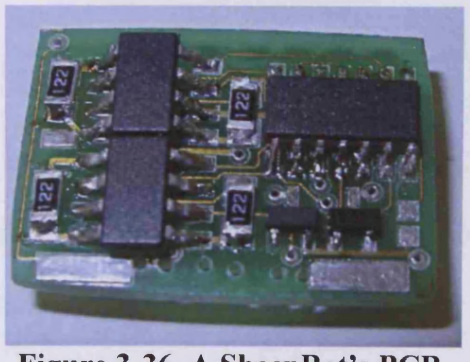


Figure 3-36. A SheepBot's PCB.

3.6.2.1 System Design

The final SheepBots will be used in the real world. According to axiom 1, the environment for the robots will be the physical world.

The robots are micro-controller based, the bodies of the robots will consist of all the hardware, and some resident software. Because the robots may be required for a number of tasks, the software part of the body will be kept minimal. It will consist of resident bootloading code and body-specific information.

The soul will be purely software. It will consist of the user program that will run on the PIC micro-controller. The soul and body have been defined in accordance with axiom 2.

The SheepBots sensors and motors are connected to the PIC input/output interface pins. Consequently, the soul will be able to interact with the body, hence fulfilling axiom 3.

The soul-body system consists of a central processing unit - the PIC, actuators (i.e. the motors), sensors and a control program. The system, once running, will be able to interact with its environment. Axiom 4 is also fulfilled.

The system is designed so that a soul can remain non-body-specific. For this to be possible, the required information specific to a body is held on that

body, in accordance with axiom 9. This information is held in the PIC's memory, as part of the software side of the body. For this implementation, the only body-specific information required is the body identification number, (*IDENT*). Axiom 5 is therefore met.

The system is also designed so that a body is not soul specific. In other words, each body can accommodate any soul, hence fulfilling axiom 6.

According to axiom 7, a trigger initiates the incarnation process. In this implementation, the incarnation process can be initiated by three different triggers: boot-up, software interrupt or hardware interrupt. At power up, the PIC waits for a program to be sent to it from a PC workstation. A specific software interrupt, for instance generated from the workstation by a simple click of the mouse, can force the PIC to download a new program. Finally, the PIC can also be configured so that a hardware interrupt triggers an incarnation, for instance a switch placed on the robot.

Axiom 8 stipulates that a trigger will initiate the de-incarnation process. This either happens at power-down, or when a new soul overwrites the previous one.

Axiom 1:	There exists an environment.	The real physical world.
Axiom 2:	Among the entities in this environment, there exist bodies and souls. Both souls and bodies can be present and contained in this environment.	A body: the entire robot hardware i.e. the chassis, motors, electronics, etc. PIC resident software. The body specific information. A soul: purely software, consisting of the user program that will run on the PIC.
Axiom 3:	A soul can interact with a body to form a system.	The SheepBots specific hardware is connected to the PIC I/O interface. The soul can directly control the hardware through that interface.
Axiom 4:	The soul-body system can interact with its environment.	The soul-body system is equipped with sensors and actuators allowing interaction with the environment.
Axiom 5:	A soul is not body specific.	A soul is free from any body specific information hence can be incarnated in any available body.
Axiom 6:	A body is not soul specific.	A body is free to be incarnated by any consistent soul.
Axiom 7:	A trigger will initiate the incarnation process. This trigger may come from the body, the soul or the environment.	The incarnation process can take place: • After powering up the PIC, • After a software interrupt • After a hardware interrupt.
Axiom 8:	A trigger will initiate the de-incarnation process. This trigger may come from the body, the soul or the environment.	The de-incarnation process occurs when the system is powered down or reset. It can also occur when a soul is overwritten by a new soul.
Axiom 9:	A body may be linked with some information, body-specific or not. The active soul may access that information.	All required body-specific information is stored on the PIC memory as part of the resident software.

Table 3-5. The SheepBot Implementation According to the Re-Embodiment Axioms.

3.6.2.2 Incarnation Process

The PIC 16F87X is fitted with both a serial communication interface (USART) and Flash memory. The Flash memory may be written in-situ without the use of a dedicated hardware PIC programmer. Consequently, a programmed received via the USART may be directly written to memory.

All the code that a PIC executes has to be in the program memory. Therefore, both the resident code handling the incarnation process and the user code will have to share the program memory. The incarnation code will reside permanently in the PIC memory in protected zone. The user code will be rewritten according to the user's needs. The combined memory map is presented in Figure **3-37**below.

The PIC was connected to a PC via a serial port. First by using a cable link and a MAX 232 to convert the signal strength from RS-232 level to TTL level (See Appendix H for schematics). Once the bootloading was operational, this connection was replaced by a two way infrared data connection based around the HSDL-1001 IrDA compliant transceivers (See Appendix H for schematics). The use of RF Transceivers was also investigated.

On power up, both the USART and a Timer are initialised. Then the PIC waits for an identification word (*IDENT*) to be sent from the PC. If the *IDENT* matches the PIC personal identification number, the PIC will acknowledge by sending back the *IDACK* word. The PIC will then receive the

program being sent. After successful completion of the transfer, the execution pointer will jump to the user program. A detailed flowchart and the source code are presented in Appendices E and F.

To further enhance the flexibility of the system, the range of signals that trigger an incarnation was extended to include peripheral interrupts and serial interrupts. Consequently, the incarnation may be triggered without having to reset the robot. A simple mouse 'click' on the PC can be used to generate a serial interrupt, thus triggering the reincarnation process. A switch or a sensor on the robot may be used to generate a peripheral interrupt.

To allow the user to have access to interrupts, the interrupt setup had to be part of the user code. However, the interrupt service routine (at least the first part of it) is part of the incarnation code. To insure that the interrupt setup remains consistent with the interrupt service routine in the Incarnation code, a template was created for the user code. This template already includes the part of the interrupt setup relevant to the incarnation code.

To allow the user to code Interrupt Service Routines (user ISR) for its own interrupt, a jump occurs at the end of the incarnation ISR to a fixed address where the user ISR must reside. A flowchart detailing interrupt handling is presented in Appendix G.

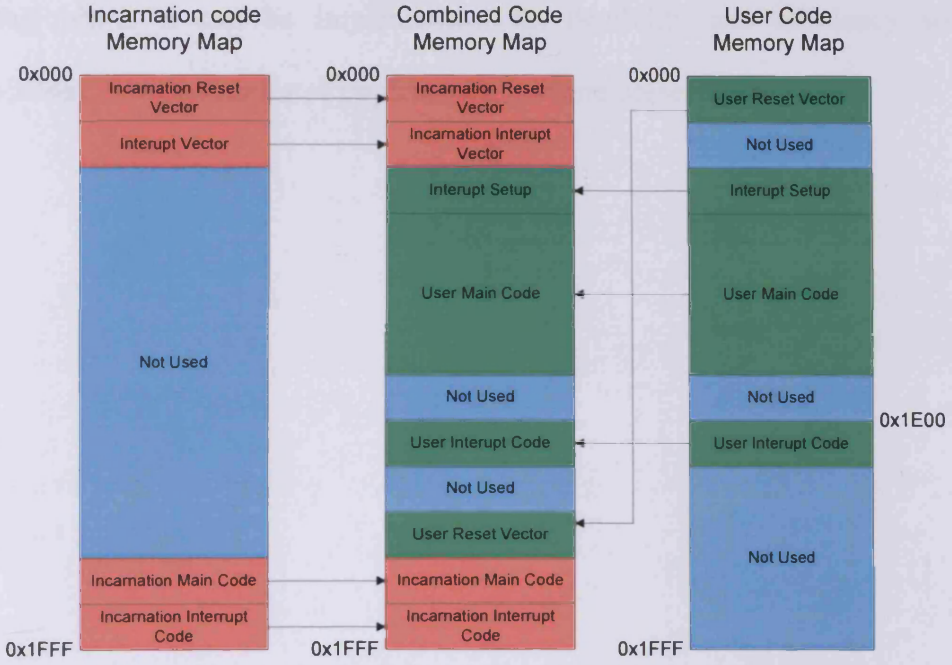


Figure 3-37. Combined Code Memory Map in the PIC.

3.7 Summary

This chapter has introduced Re-embodiment through a set of axioms along which it can be implemented. Its flexibility and efficiency were discussed. Two examples of implementation were presented.

Chapter 4 A New Mono Vision Distance Measurement Method

4.1 Preamble

It has been established that one of the basic requirements of embodiment, according to the definition by Quick *et al.*, is that there exists perturbatory channels between a system and its environment **[Quick *et al.*, 1999a; Quick *et al.*, 1999b]**. In robotics, these perturbatory channels are instantiated by the use of actuators and sensors. A robot may use its actuators to perturb its environment state. In return, through its sensing abilities, the robot may be perturbed by the environment. Furthermore, it has been established that there are several degrees of embodiment **[Werger and Mataric, 1999]**. Quick et al. state that the extent of the sensory capabilities of a robot has a direct impact on the extent of the embodiment of the system in an environment **[Quick and Dautenhahn, 1999]**. They therefore suggest that the better the sensory resources of a robot are, the greater its interacting capabilities with the environment will be.

Sensory capabilities are primordial in robotics **[Borenstein *et al.*, 1996; Adams, 2002]**. The classical AI approach in robotics failed mainly because of sensor unsuitability through noise and inaccuracy **[Fikes *et al.*, 1972; Nilsson,**

1984]. Images and sequences of images potentially carry a tremendous amount of information about the environment **[Criminisi, 1999]**.

This chapter presents a new monocular distance measurement method. Firstly, existing methods for distance measurements are reviewed with a particular emphasis on vision based ones. Secondly, the geometrical projection model on which the new measurements method is based is detailed. Thirdly, a series of experimental tests are presented, highlighting accuracy and performance. Finally, two applications are presented, one in which robots follow each other in a simulated environment, and a second one where real robots track and kick a ball.

4.2 Review of Related Work

4.2.1 Existing Distance Measurement Methods

There already exist a number of methods to measure distances in the physical world. Those applicable to mobile robotics can be divided into two categories: Active and passive methods. Active devices emit signals into the environment and receive them back, whereas passive devices do not **[Borenstein *et al.*, 1996; Adams, 2002]**.

4.2.1.1 Active Methods

Ultrasonic

Most active distance measurement methods are based on the Time-Of-Flight principle (TOF). In TOF systems, a pulse of signal is emitted into the

environment. Then, using elementary physics, the distance is determined by multiplying the velocity of the energy wave and the time required to travel the round-trip distance. Many TOF distance measurement systems use ultrasonic technology for emitting and receiving energy waves. Systems using Ultrasound transducers (or RF transducers) can provide a relatively cheap solution to distance measurements **[Borenstein *et al.*, 1996]**.

The advantages of TOF systems arise from the direct nature of their straight-line active sensing. The returned signal follows essentially the same path back to a receiver located coaxially with or in close proximity to the transmitter. In fact, it is possible in some cases for the transmitting and receiving transducers to be the same device. The absolute range to an observed point is directly available as output with no complicated analysis required, and the technique is not based on any assumptions concerning the planar properties or orientation of the target surface. Furthermore, TOF sensors maintain range accuracy in a linear fashion as long as reliable echo detection is sustained **[Borenstein *et al.*, 1996]**.

Potential inaccuracy in TOF systems can arise from a number of points. Firstly, the speed of propagation of the signal may vary in the environment. This is particularly true in the case of acoustical systems, where the speed of sound is markedly influenced by temperature changes, and to a smaller extent by humidity. Secondly, there may be uncertainties when determining the exact time of arrival of the reflected pulse. Detection uncertainties errors are caused by the wide dynamic range in returned signal strength due to varying

reflectivity of target surfaces. These differences in returned signal intensity influence the rise time of the detected pulse, and in the case of fixed threshold detection will cause the more reflective targets to appear closer. Thirdly, ultrasound sensors can be confused by the interaction of the incident wave with the target surface. When radio waves strike an object, any detected echo represents only a small portion of the original signal. The remaining energy reflects in scattered directions and can be absorbed by or pass through the target, depending on surface characteristics and the angle of incidence of the beam. Instances where no return signal is received at all can occur because of specular reflection at the object's surface, especially in the ultrasonic region of the energy spectrum. If the transmission source approach angle meets or exceeds a certain critical value, the reflected energy will be deflected outside of the sensing envelope of the receiver. In cluttered environments sound waves can reflect from (multiple) objects and can then be received by other sensors. This phenomenon is known as crosstalk. Ultrasonic transducers themselves have several downfalls. They only give the distance to the first encountered obstacle within their operating volume. Furthermore, it is difficult to know what that operating volume is and what the sensor response across that volume will be. Finally, they are unreliable at detecting smaller objects **[LoPresti *et al.*, 2002]**.

Many successful applications in robotics use ultrasonic sensors in tasks such as obstacle avoidance, map building, localisation or wall following **[Zou *et al.*, 2000; Palacin *et al.*, 2003]**. There are a number of commercially

available sensors such as the SRF04 and SRF08 ultrasonic range finder designed by the British company Devantech Ltd **[Devantech, 2004]**, or the more established Polaroid sensors **[Cao and Borenstein, 2002]**. Commercial robots, such as the Pioneer mobile robot **[Gerkey and Mataric, 2002]**, are often fitted with such a type of sensor. Further applications in the real world include systems on car bumper to ease parking **[Siuru, 1994; Adams, 2002]**.

Laser Range Finder

Another approach to measuring distance is based on the use of laser range finders. The distance is derived by either TOF or phase analysis of the laser beams **[Borenstein *et al.*, 1996; Adams, 2002]**.

Laser-based TOF distance measurement systems, (also known as LASER radar or LIDAR) first appeared in work performed at the Jet Propulsion Laboratory, Pasadena, in the 1970s **[Lewis and Johnston, 1977]**. Laser energy is emitted in a rapid sequence of short bursts aimed directly at the object being ranged. The time required for a given pulse to reflect off the object and return is measured and used to calculate distance to the target based on the speed of light **[Adams, 2002]**.

Potential sources of error in this type of TOF systems include uncertainties in determining the exact time of arrival of the reflected pulse and inaccuracies in the timing circuitry used to measure the round-trip time of flight. The propagation speed of the signal in the environment will dictate the required response time of the timing circuitry to achieve reasonably accurate

measurement. In applications where light is used as the active signal, the timing demands are severe. Typically a desired resolution of one millimetre requires a timing accuracy of three picoseconds ($3\times10^{-12}s$) **[Borenstein *et al.*, 1996]**. As a result TOF LIDAR systems are relatively expensive due to the high speed and precision of the electronics necessary for timing the pulse transmission-reception time.

TOF LIDAR are commercially available as ready-made systems. The SICK sensor has been the most popular in the field of mobile robotics **[Wetteborn, 1993; Adams, 2002]**. It is available on commercial robots such as the Pioneer platform **[Gerkey and Mataric, 2002]**. It has been successfully used in a number of applications such as motion planning and maps building **[Tovar *et al.*, 2002]**.

The phase-shift measurement (or phase-detection) ranging technique involves continuous wave transmission as opposed to the short pulsed outputs used in TOF systems. The reflected signal is then analysed using method such as frequency modulated continuous wave (FMCW) or amplitude modulated continuous wave (AMCW). In, FMCW approach, the beat frequency between an FMCW laser signal and its reflection is measured. For close range applications and particularly in mobile robotic applications, a simple means of determining range is by measuring the phase shift between an AMCW and its received reflection. However, there can be more than one distance corresponding to any given phase shift measurement. The potential for erroneous information as a result of this ambiguity reduces the appeal of phase-

detection schemes. Some applications simply avoid such problems by arranging the optical path so that the maximum possible range is within the ambiguity interval. Alternatively, successive measurements of the same target using two different modulation frequencies can be performed **[Borenstein *et al.*, 1996]**. As TOF systems, phase shift systems suffer from uncertainties in determining the exact time of arrival of the reflected pulse, and are influence by the reflective property of the target. They are also relatively expensive. Although less popular, they have been used for applications such as world modelling **[Mettenleiter *et al.*, 2000]**.

Optical Triangulation

Optical triangulation position sensors use reflected waves, whose source may be an LED, infrared, or laser. Light emitted from the source is reflected off any object in the field of view and returned to a sensor. This creates a triangle between the points of reflection, the emitter and the detector. A precision lens transmits the reflected light onto various portions of the sensor. The sensor, usually integrally housed with the emitter, detects the angle at which the beam is reflected and calculates a distance to the obstacle. An output that varies with the distance is then provided. The sensing part of optical triangulation position sensors is usually a Charged Coupled Device (CCD).

These sensors are relatively cheap and readily available. For instance, the Sharp sensors are proving increasingly popular in robotics. However, with such sensors, the accuracy falls off rapidly with increasing range, and the depth of

field (minimum to maximum measurable distance) is typically limited **[LoPresti *et al.*, 2002]**.

Structured Light

Other active methods to derive distance measurement is structured light. Such devices employ cameras to acquire images of an object illuminated by a regular light pattern. An auxiliary devices project a light pattern or a set of patterns onto an object. The shape of the object, as well as the distance from the sensor to the object can then be computed from the deformation of the projected grid. Structured light-based approaches have been used for accurate measurement of objects **[Li and Chen, 2003]**. However, the need for auxiliary light projection devices leads to a severe loss in the flexibility of the measuring tool.

4.2.1.2 Passive Methods

Active distance measurement sensors are generally affected by interference with the active signal. On the other hand, passive devices such as cameras do not suffer from the above problems and can be appropriate for a wider range of applications. Images potentially carry much more information than raw detectors from any of the sensors reviewed so far **[Criminisi, 1999]**. Cameras return bi-dimensional data, rather than mono-dimensional ones. They also have a relatively dense sampling across their field of view. As a result, they can be used to quantify a number of different parameters in addition to

distance measurement with the same data source. However, taking measurements of the world from images is complicated by the fact that in the imaging process the 3D space is projected onto a planar surface, with some unavoidable loss and distortion of information **[Criminisi, 1999]**.

Stereo Systems

Generally, most of the research to date relies on stereovision, where two image of the same scene are captured from different viewpoints **[Faugeras, 1992]**. The distance in the scene computed by triangulation between the locations of matching features in images. Correspondence between features is usually established by searching through the image and comparing local pixel neighbourhood using measures such as Sum of Square Differences (SSD) or Sum of Absolute Difference (SAD) **[Brown *et al.*, 2003]**. Note that a full review of the massively rich field of stereovision falls beyond the scope of this thesis. An excellent examination of the progress in the field as been put together by Brown *et al.* **[Brown *et al.*, 2003]**. Instead, this section will concentrate on providing basic principles and highlighting pro and cons.

Although passive stereovision is one of the oldest research topics in computer vision, its use in robotics was at first limited by the large amount of computational power required. The first system that came close to achieving frame-rate was reported by Webb in 1993 **[Webb, 1993]**. He implemented the multi-baseline stereo algorithms of Kanade et al. **[Kanade *et al.*, 1992]**, on the Carnegie Mellon University (CMU) Warp machine **[Crisman and Webb,**

1991]. Three images were used for this system. For efficiency, the sum of SAD (SSAD) was implemented. Sixty-four 'iWarp' processors were used to achieve fifteen frames per second with 256x240 pixel images.

In the past decade, swift advances in affordable computational power have opened new possibilities. Real-time stereo processing has become accessible to common desktop computers. For instance, Point Grey's Triclops **[Kimura *et al.*, 1999]** runs at twenty frames per seconds for 320x240 pixel images on a 1.4 GHz Pentium IV machine. Likewise, SRI's Small Vision Module (SVM) now runs at 30 fps for 320x240 pixel images on a 700 MHz Pentium III **[Konolige, 1997]**. With inexpensive and compact real-time systems like these now commercially available, many applications that previously were impractical can now be explored. However, these computational requirements remain relatively high for robotic applications, especially in multi robot systems.

Beyond the computational requirement issue, stereovision suffers from limitation due to the triangulation geometry. A compromise has to made between high image resolution and reduced ambiguity in matching **[Hebert, 2000]**. Although systems with three or more cameras attempt to address this problem, by integrating matching result from multiple images, it remains an issue in contemporary robotic research.

Another downfall to stereovision systems is their sensitivity to points that are visible to one camera only. Detecting these occlusions **[Wildes, 1991]** and

compensating for their occurrence remains a main focus of research **[Zitnick and Kanade, 2000]**.

Monovision

As opposed to stereovision, the amount of research done on monovision system has been much less substantial. One of the main objectives of computer vision research has been to reconstruct three-dimensional (3D) scenes. Generally, monovision has been seen as being too restrictive in the amount of information that it can provide for scene reconstruction. However, some metric quantities can be computed with extra constraints.

Some algorithms compensate for the fact that only one camera is available by collecting a series of two or more images to extract useful information or reconstruct 3D scenes. Peer and Solina **[Peer and Solina, 2002]** developed a panoramic depth imaging system. In their system, a single camera rotates in an environment and assembles the captured images into a mosaic. An offset between the camera focal point and the centre of rotation allows for depth maps of the environment to be computed. Reconstruction, based on the system geometry and a symmetric pair of stereo panoramic images, can also be achieved.

Another possibility is to analyse object shadows. This was successfully applied to ball tracking in sport events. For instance, Kim et al **[Kim *et al.*, 1998]** compute the position of a ball from single images of a football game. They make use of the shadows on the ground plane to track the ball.

Further information may also be extracted through texture analysis. Cantoni et al. derive depth from monocular images through histogram inspection **[Cantoni *et al.*, 2001]**. They observed that on images, distant elements are less clearly defined and that the colours of different objects tend to blend, whereas edges of element in the foreground are much more clearly defined. They also observed that this was more pronounced if haze or fog was present, or when underwater images were examined. They explain this phenomena by the fact that rays of light are diffused by molecules of opaque medium. Air contains a great number of water participles, which refract light. As a result, in monochromatic images, the background grey levels tend to be a weighted average of all the grey levels present in the image itself, thus explaining the fact that images become more blurry as the distance from the observer increases. Note that this blur is not to be mistaken for the one produced by camera lenses. As a result for their algorithm to produce reliable data, they rely on defocusing blur to be avoided.

Different methods to derive depth from single images based on focus and blur information have emerged in the literature: Depth from Automatic Focusing (DFAF), Depth from Defocusing (DFD) and Depth from Automatic Defocusing (DFAD) **[Aslantas and Tunçkanat, 2003; Deschênes *et al.*, 2004]**.

DFAF methods attempt to find the sharpest image possible of an object by varying camera parameters. After a sharp image has been obtained, depth

can be computed using optics calculation. **[Krotkov, 1987; Nayar and Nakagawa, 1990; Aslantas, 2001]**.

DFD methods do not require an element to be in focus in order to compute its depth. If an image of a scene is acquired by a real lens, points in the scene at a particular distance from the lens will be in focus, whereas points at other distances will be out of focus by varying degrees depending on their distances. Using this information, the depth of an object can be determined **[Grossman, 1987; Aslantas and Tunçkanat, 2003]**. This even led to successful application to robotics for obstacle avoidance and to detect steps **[Nourbakhsh *et al.*, 1997]**.

DFAD is a combination of DFD and DFAF. The depth is computed by making use of both the blur information, and some alteration to the camera settings. However, a sharp image of the object is not required **[Aslantas, 1997]**.

Another possible route is to compute depth from perspective effects. This usually involves determining vanishing points from perspective images **[Almansa *et al.*, 2003]**. Then spatial information can be extracted from images **[Criminisi, 2001]**. For instance, Se and Brady developed an algorithm to detect features for navigation **[Se and Brady, 2003]**. Criminisi derived new methods to measure height on un-calibrated images **[Criminisi *et al.*, 2001]** and reconstruct 3D scenes from historical paintings **[Criminisi, 1999]**.

Remondino and Roditakis devised a system to reconstruct a human skeleton from a single un-calibrated image. They first simplify orthographic projection equation using a scaling factor that mimic the effect that the image

of an object shrinks with the distance. They then assume that the depth of the observed object is small compared to the distance between the camera and the object, thus implying that their scaling factor remains virtually constant. Then, knowing the physical length of a segment on the image and the depth of that segment, they can compute distance on the image and recreate a 3D scenes from singles images.

Lenser and Veloso put forward an interesting monocular obstacle avoidance method **[Lenser and Veloso, 2003]**. Their system copes with known and unknown obstacles by detecting occlusions of a floor of known colour. Range and angle to the objects are calculated and used to create a radial model of the robot's vicinity. This modelling component keeps track of objects that are currently outside the field of view of the camera allowing the robot to avoid obstacles it is not currently looking at.

All those methods rely on extra constraints to compensate for the fact that only one camera or image is available. The method presented in this chapter will rely on two prerequisites. The first assumption is that the ground of environment, where the robots evolve, is flat. The second is that the camera, capturing the images, may be calibrated to obtain some internal in external information.

4.3 Monocular Distance Measurement

4.3.1 Image Formation and Camera model

4.3.1.1 Perspective Projection

The pinhole camera model is the most general and well-known linear camera model. This system consists of two screens or planes; see Figure 4-1. The first one, the focal plane F, has an infinitesimally small hole. This hole O is also known as the optical centre or focal point. Light reflected on objects passes through this hole to form an inverted image on the second plane. The second screen is referred to as the image or retina plane R. The distance between the two planes is the focal length f. The line going through the optical centre O and perpendicular to the focal plane F is known as the focal axis or principal axis. The focal axis pierces the image plane at the principal point o. The model is often simplified by placing the retinal plane R between the focal point O and the object, so that the image is not inverted; see Figure 4-2. This mapping between a three-dimensional object or scene and a two dimensional plane is called perspective projection or central projection.

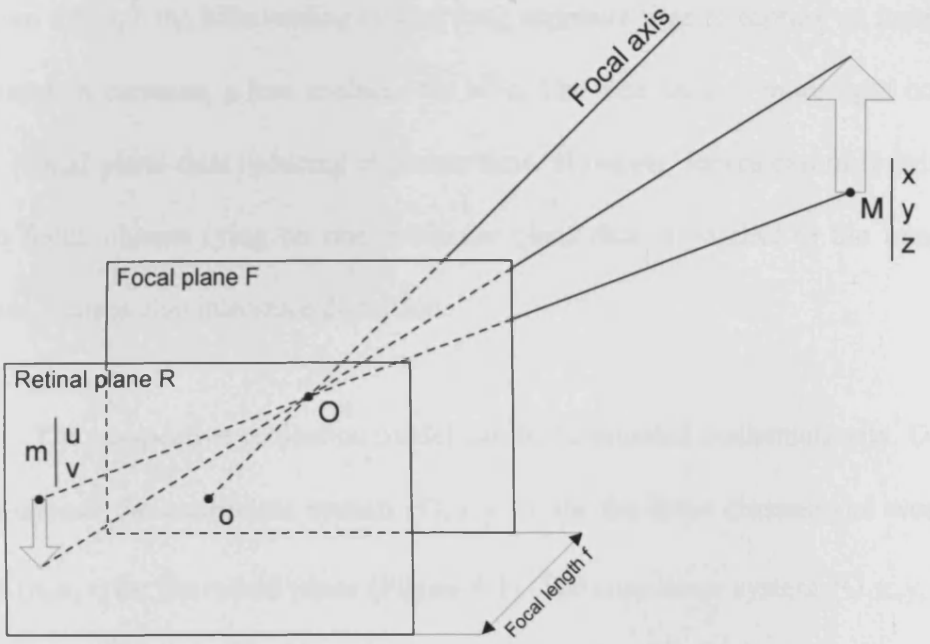


Figure 4-1. Perspective Projection in Pinhole Camera.

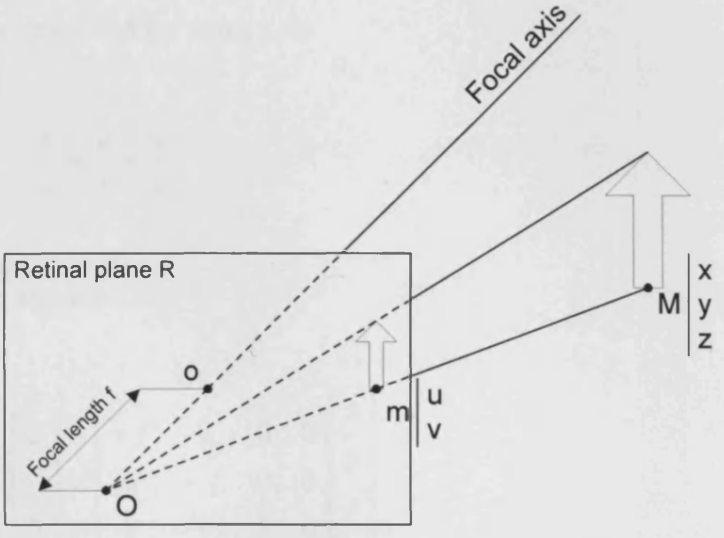


Figure 4-2. Simplified Perspective Projection.

In a real application, the use of a pinhole is unpractical. Very little light passes through the hole leading to very long exposure time to capture an image. In modern cameras, a lens replaces the hole. The lens focuses more light onto the retinal plane thus reducing exposure time. However, lenses can only bring into focus objects lying on one particular plane that is parallel to the image plane. Lenses also introduce distortion.

The perspective projection model can be formulated mathematically. One can choose the coordinate system (O,x,y,z) for the three dimensional world and (o,u,v) for the retinal plane (Figure 4-1). The coordinate system (O,x,y,z) located at the optical point O, with the z axis perpendicular to the focal plane F, is often called the standard coordinate system of the camera, or camera reference frame. The relationship between the image coordinate and the three dimensional space can be written as:

$$-\frac{f}{z}=\frac{u}{x}=\frac{v}{y} \tag{4.1}$$

This can be rewritten as:

$$\begin{bmatrix} U \\ V \\ S \end{bmatrix} = \begin{bmatrix} -f & 0 & 0 & 0 \\ 0 & -f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{4.2}$$

The interesting point about the vector $[U \quad V \quad S]^T$ is that:

$$\frac{U}{S} = u, \frac{V}{S} = v \quad \text{if and only if } S \neq 0 \tag{4.3}$$

Note that Equation **(4.2)** is defined up to an arbitrary scale factor. This Equation expresses the relationship between retinal (or image) coordinates and space (or world) coordinates in linear matrix form, such that:

$$m = PM \tag{4.4}$$

where the 3x4 homogeneous matrix P is the projection matrix. The retinal and space coordinates are represented by the homogeneous vector $m = [U \quad V \quad S]^T$ and $M = [X \quad Y \quad Z \quad 1]^T$

4.3.1.2 Intrinsic and Extrinsic Camera Parameters

The camera model is completely specified once the matrix P is determined. Thus P must include all the camera parameters. Physical camera parameters are commonly divided into extrinsic and intrinsic parameters. The extrinsic and intrinsic parameters can be expressed in matrix form, such that:

$$P = P_{\text{int}} P_{ext} \tag{4.5}$$

Extrinsic parameters define the location and orientation of the camera reference frame with respect to a known world reference frame. They are the translation T and Rotation R, which specify the transformation between retinal and world reference. They are defined as follows:

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \quad \textbf{(4.6)}$$

$$T = \begin{bmatrix} t_1 & t_2 & t_3 \end{bmatrix}^T \quad \textbf{(4.7)}$$

Matrix R and vector T can be combined into a matrix to form P_{ext}:

$$P_{ext} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \quad \textbf{(4.8)}$$

The intrinsic parameters are independent of the camera position and orientation in space. They are necessary to link the pixel coordinates of an image point with the corresponding coordinates of the camera reference frame. They characterise the optical, geometrical and digital attributes of the camera. They usually include:

- The focal length f.
- The location of the principal point in pixel coordinates (o_x, o_y).

- The effective horizontal and vertical pixel sizes s_x and s_y. These allow for change from metric units to pixels and vice versa. Note that here, as usual in computer vision literature, the origin of the image coordinate system is in the upper left corner of the image array.
- The skew coefficient α_c defines the angle between the x and y pixel axes, such that $\alpha_c = 0$ when x and y are perpendicular.

$$P_{\text{int}} = \begin{bmatrix} -f/s_x & f\alpha_c & o_x \\ 0 & -f/s_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \quad \textbf{(4.9)}$$

The components of the intrinsic matrix $-f/s_x$ and $-f/s_y$ represent the focal length, a unique value, expressed in units of horizontal and vertical pixels. To facilitate the notation, they can be replaced by f_x and f_y respectively.

$$P_{\text{int}} = \begin{bmatrix} f_x & f\alpha_c & o_x \\ 0 & f_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \quad \textbf{(4.10)}$$

The extrinsic and intrinsic parameters can be inserted in the linear matrix perspective projection equation (Equation **(4.4)**) to form:

$$m = P_{\text{int}} P_{ext} M \quad \textbf{(4.11)}$$

4.3.1.3 Image Distortion

Image distortion, as introduced by the lens and the camera assembly is a long known phenomenon **[Clarke and Fryer, 1998]**. It is particularly evident at the image periphery and worsened by large field of view. Although it does not significantly influence the image quality, it considerably affects the image geometry. The distance measurement algorithm presented in this thesis is based on the analysis of the image geometry. It is therefore important, for the accuracy of the measure, to be able to work with undistorted images. Therefore, the distortion needs to be modelled by a set of coefficients. The camera calibration will need to determine those coefficients. Then undistorted images will be computed from original captured images and the distortion coefficients.

Image distortion is due to a number of different phenomena; it is usually modelled by a radial and tangential component. Radial distortion is mainly due to spherical distortion of the length, which creates a slightly curved focal plane. This introduces both geometrical and focal distortion: Points are distorted geometrically from the true perspective position, by being displaced radially in the image plane **[Heikkila and Silven, 1997]**. Points also become more blurred as they approach the edge of the image. The radial distortion can be approximated. Given a point p_n of coordinates $[u_n \quad v_n]^T$ in the retinal plane as derived from the pinhole projection.

After radial distortion, the new coordinate of the point $[u_{dr} \quad v_{dr}]^T$ can be expressed as follow:

$$\begin{bmatrix} u_{dr} \\ v_{dr} \end{bmatrix} = (1 + k_1 r_n^2 + k_2 r_n^4 + k_3 r_n^6 + ...) \begin{bmatrix} u_n \\ v_n \end{bmatrix} \tag{4.12}$$

where $k_1 ... k_n$ are radial distortion coefficients and:

$$r_n^2 = u_n^2 + v_n^2 \tag{4.13}$$

Tangential distortion is mainly due to imperfect centring of the lens components and other manufacturing defects in a compound lens. Tangential distortion can be approximated as follows:

$$\begin{bmatrix} x_{dt} \\ y_{dt} \end{bmatrix} = \begin{bmatrix} 2p_1 u_n v_n + p_2(r_n^2 + 2u_n^2) \\ p_1(r_n^2 + 2v_n^2) + 2p_2 u_n u_n \end{bmatrix} \tag{4.14}$$

where p_1 and p_2 are the tangential distortion coefficients. The total distortion modelled so far can be expressed as

$$\begin{bmatrix} x_d \\ y_d \end{bmatrix} = \begin{bmatrix} x_{dr} \\ y_{dr} \end{bmatrix} + \begin{bmatrix} x_{dt} \\ y_{dt} \end{bmatrix} \tag{4.15}$$

An accurate camera model can be derived by combining the pinhole projection with the correction for the radial and tangential distortion components.

4.3.1.4 Camera Calibration

A priori requirement of the distance measurement method presented in this thesis is that it is possible to calibrate the camera to derive the required intrinsic and extrinsic camera parameters, as well as the distortion coefficients.

A number of visual algorithms have been developed to compute intrinsic and extrinsic camera parameters. This procedure is called camera calibration. Usually, calibration algorithms assume some of the camera internal parameters to be known and derive the remaining ones. Common assumptions are: unit aspect ratio, zero skew or coincidence of principal point and image centre.

The calibration techniques described in this section are based on a series of single view. The work of Tsai **[Tsai, 1987]** has been one of the most popular in the field of camera calibration. From a number of single images of a known, planar, calibration grid it estimates the focal length of the camera, its external position and orientation assuming known principal point. An attempt to analyse the accuracy of the estimated parameters is also reported.

Caprile and Torre, in their classical work **[Caprile and Torre, 1990]**, develop an algorithm to compute the internal and external parameters of the camera from single views, assuming unit aspect ratio and zero skew. They make use of simple properties of vanishing points. These can be extracted directly from the image by intersecting images of parallel lines. A simple calibration device consisting of a cube with sets of parallel lines drawn on its faces is used. The authors demonstrate that the principal point of the camera coincides with the orthocentre of the triangle whose vertices are the three vanishing points for three orthogonal directions.

The problem of camera calibration is also discussed by Faugeras **[Faugeras, 1993]**. Algorithms to compute the projection matrix and eventually the camera internal parameters from only one view of a known 3D grid are presented. Linear and non-linear methods for estimating the projection matrix are analysed, as well as the robustness of the estimate and the best location for the reference points.

In Liebowitz and Zisserman's work **[Liebowitz and Zisserman, 1998]** camera self-calibration is obtained simply from images of planar structures like building facades or walls, with distinguishable geometric features. Use is made of scene constraints such as parallelism and orthogonality of lines and ratios of lengths. No specifically designed calibration object is required.

For the purpose of the work presented in this thesis a calibration procedure was implemented to determine the intrinsic and extrinsic parameters of the robot's camera. The radial and tangential distortion coefficients are also determined using this implementation. This calibration procedure is based on the calibration algorithm developed by Zhang **[Zhang, 1999; Zhang, 2000]**. The first step involves finding the homography for all points in the series of images. The intrinsic parameters are then initialised and the distortion is set to zero. The extrinsic parameters for each image are then computed. Finally, an optimization procedure is applied to minimizes the error of projection points with all the parameters. This method yields an estimation of the camera intrinsic parameters (assuming zero skew) and, for each image, the corresponding extrinsic parameters. The algorithm assumes the bottom left corner of the chess board to be the origin of the world coordinate frame, thus the different extrinsic parameters for each images. However, only the extrinsic parameters of the camera with respect to the robot frame are of interest. In order to derive those, it is arranged so that one for the images the location and orientation of the chessboard with respect to the robot frame are known.

4.3.2 Measurement From a Single View

4.3.3 Prerequisite Statement

To harmonize all spatial representations the robot reference frame (R,X,Y,Z) is defined to be such that:

- The X axis is aligned with the driving wheel axis.
- The Y axis is vertical, perpendicular to the environment floor.
- The Z axis is pointing forward.
- The origin R of the robot reference frame is at equal distance from each wheel.

All distance measurements will be made in this reference frame from the origin.

4.3.4 Measurement Model

The basic measurement model will allow for the distance between the robot and an object in contact with the floor to be estimated. It is assumed that:

(i) The extrinsic and intrinsic camera parameters are known in the robot reference frame.

(ii) The environment floor is planar.

(iii) The object is in contact with the floor at a point M.

(iv) The coordinate m, of the projection of M on the focal plane, can be estimated.

Based on the projection model and the above assumptions the equation for the measurement model can be derived. Given the perspective projection equation (Equation **4.16**), the point $M=[x \quad y \quad z \quad 1]^T$ in robot space, the coordinates $m=[u \quad v]^T$ of its projection on the image plane can be computed.

$$\begin{bmatrix} U \\ V \\ S \end{bmatrix} = \begin{bmatrix} f_x & 0 & o_x \\ 0 & f_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} x \\ y \\ x \\ 1 \end{bmatrix} \tag{4.16}$$

Knowing that $\frac{U}{S}=u$ and $\frac{V}{S}=v$, if and only if $S \neq 0$, one can rewrite the above as:

$$u = \frac{(r_{11}x + r_{12}y + r_{13}z + t_1)}{(r_{31}x + r_{32}y + r_{33}z + t_3)} f_x + o_x \tag{4.17}$$

and

$$v = \frac{(r_{21}x + r_{22}y + r_{23}z + t_2)}{(r_{31}x + r_{32}y + r_{33}z + t_3)} f_y + o_y \tag{4.18}$$

The intrinsic parameters of the camera, its extrinsic parameters in the robot frame and the coordinates of the projected point in the image frame are known. The y component of the coordinate of M can be derived, knowing that M is in contact with the floor and that the distance from the origin R, of the robot frame, to the floor, should be obtainable from the robot's specification. This

leaves a system of two equations (**4.17** and **4.18**) with two unknowns: x and z.

$$\begin{cases} u = \dfrac{(r_{11}x + r_{12}y + r_{13}z + t_1)}{(r_{31}x + r_{32}y + r_{33}z + t_3)} f_x + o_x \\ v = \dfrac{(r_{21}x + r_{22}y + r_{23}z + t_2)}{(r_{31}x + r_{32}y + r_{33}z + t_3)} f_y + o_y \end{cases} \quad \textbf{(4.19)}$$

The equation system **4.19** can be rewritten as:

$$\begin{cases} x((u-o_x)r_{31} - f_x r_{11}) + z((u-o_x)r_{33} - f_x r_{13}) \\ + y((u-o_x)r_{32} - f_x r_{12}) + ((u-o_x)t_3 - f_x t_1) = 0 \\ \\ x((v-o_y)r_{31} - f_y r_{21}) + z((v-o_y)r_{33} - f_y r_{23}) \\ + y((v-o_y)r_{32} - f_y r_{22}) + ((v-o_y)t_3 - f_y t_2) = 0 \end{cases} \quad \textbf{(4.20)}$$

The Equation system **4.20** is equivalent to a simple equation system of the generic form

$$\begin{cases} a_1 X + b_1 Z + c_1 = 0 \\ a_2 X + b_2 Z + c_2 = 0 \end{cases} \quad \textbf{(4.21)}$$

with solutions

$$x = \frac{-b_1(a_2c_1 - a_1c_2)}{-a_1(a_1b_2 - a_2b_1)} - \frac{c_1}{a_1} \quad \textbf{(4.22)}$$

and

$$z = \frac{a_2 c_1 - a_1 c_2}{a_1 b_2 - a_2 b_1} \tag{4.23}$$

where

$$\begin{aligned} a_1 &= (u - o_x) r_{31} - f_x r_{11} \\ b_1 &= (u - o_x) r_{33} - f_x r_{13} \\ c_1 &= y\left((u - o_x) r_{32} - f_x r_{12}\right) + (u - o_x) t_3 - f_x t_1 \\ a_2 &= (v - o_y) r_{31} - f_y r_{21} \\ b_2 &= (v - o_y) r_{33} - f_y r_{23} \\ c_2 &= y\left((v - o_y) r_{32} - f_y r_{22}\right) + (v - o_y) t_3 - f_y t_2 \end{aligned} \tag{4.24}$$

4.3.5 Basic Centreline Measurement

From the measurement model formulated above, distances from robot to object can be derived in a number of ways. The simplest of those would be to set the physical location and orientation of the camera such that the distance from the robot to the object is directly equal to z as defined in equation **4.23**. For instance, in the case where the focal axis of the camera lies in the plane defined by the Y and Z axes of the robot frame (R, X, Y, Z). In this particular situation, distances to objects directly in front of the robot can be computed by extracting the point at the intersection of the y axis of the camera frame (i.e. the vertical line on the image passing through the optical centre O) and the object edge in contact with the floor.

4.3.6 Measurements Across the Field of View

The basic centreline measurement allowed for distances to objects straight in front of the robot to be computed. In some applications, measurement to objects across the field of view may be required. In those cases distance d to object can be computed as follows:

$$d = \sqrt{x^2 + z^2} \tag{4.25}$$

with x and z as defined in equations **4.22** and **4.23**. With this method, object edges in contact with the floor may be fully evaluated to retrieve object properties such as orientation. This can also allow, with tracking over time, for object velocities and changes in orientation to be assessed.

4.3.7 Off-the-Ground Objects

So far consideration has been limited to measurement to objects that are in contact with the floor. The measurement method can now be extended to objects or features that are not in contact with the floor. In this particular case it will be assumed that the distance between the detectable feature on the object and the floor is known. This required priory information may at first seem to restrict the number of possible applications. It is true that knowing the height from the floor to features on objects in an environment seems unrealistic for robot applications. However knowing the height of particular features on other robots within the MRS would appear perfectly reasonable. This in fact could

make the implementation easier. One could carefully choose these features for each robot to ensure that precise detection is made easier. An easy-to-detect colour against a contrasting background would even allow for sub-pixel edge detection. Then using the same formulation as above, distance to object can be computed.

4.3.8 Accuracy Estimation

Now that formulation of the distance measurement has been derived, it is important to theoretically assess its accuracy and limitations. The measurement primarily relies on two sources of information: The camera parameters and the location of the object feature on the image. Inaccuracies and noise in this data will result in imprecision in the distance measurement. First, the effect of feature detection in the image was studied. More specifically the consequence of non continuous feature detection across the image due to finite pixel size was investigated. Arguably this effect will be different for every possible extrinsic and intrinsic camera parameter. Nevertheless, and in order to gain a better understanding for such effects, an analysis was conducted with an arbitrary set of parameters. The results are presented in Figure 4-3. As one may expect, the inaccuracy varies greatly across the range. In that particular example, a single pixel accounts for all distances from thirty-six meters to infinity. However at the other end of the range, a pixel covers just over a millimetre.

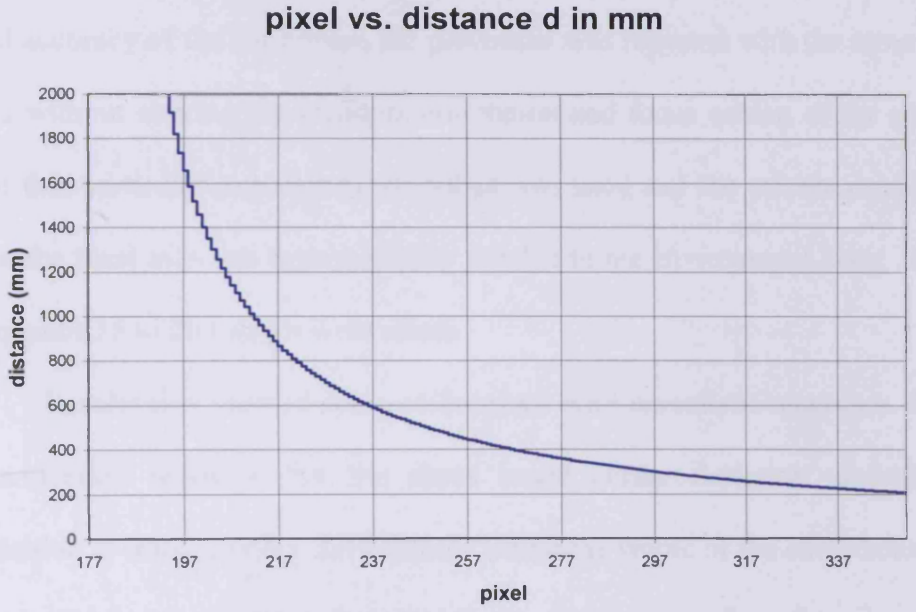


Figure 4-3. The Effect of Discrete Pixels on the Distance Measurement

4.4 Experimental Measurements and Analysis

4.4.1 Calibration

For each available robot in the BOM fleet, the camera was calibrated a number of times throughout this research. To formally assess the repeatability and accuracy of the calibration the procedure was repeated with the same robot and without altering the physical orientation and focus setting of the camera. For this particular experiment, Robobug was used and the camera oriented so that the focal axis was approximately parallel to the environment floor. In each sequence 15 to 20 images were taken.

Results first showed some calibrations with unrealistic outcomes. Closer examination revealed that the chess board corner detection algorithm is sensitive to image quality. In instances where the whole of the chess board was not in the image, or if the chess board was too far away from the camera, not all corners were detected. In instances where one or more images were blurry in the calibration sequence, results were also inconsistent. Eventually consistent results were achieved through careful screening of every image in each calibration sequence. They are presented in Table 4-1.

Calibration Nb	fx	fy	Ox	Oy
1	443.4981	485.8984	160.6691	150.0001
2	443.8726	484.6134	160.9896	150.1939
3	444.5847	483.7013	160.9767	149.9312
4	443.3297	485.0668	161.1719	150.2559
5	444.1101	483.5372	160.8502	150.2763
6	443.8022	484.2975	161.1922	149.8541
7	443.2476	483.7353	160.9558	149.6146
8	443.6057	484.2948	160.6625	149.9808
9	444.8007	485.2111	161.0576	149.5411
10	444.2613	484.4907	161.1664	149.2969
Average	443.9113	484.4846	160.9692	149.8945
Min	443.2476	483.5372	160.6625	149.2969
Max	444.8007	485.8984	161.1922	150.2763

Table 4-1. Typical Camera Calibration Results

4.4.2 Distance Measurement

Following the theoretical accuracy assessment, the measurement method also had to be assessed in situ. Again, precision will vary with different camera parameters. Nevertheless, to gauge the performance of the measurement system an experiment was set up. Once more, Robobug was used, and the camera was orientated such that the optical axis lies parallel to the environment floor.

A series of seventeen measurements was taken at different distances. Figure 4-4 shows the nominal values against the measured values. Figure 4-5 shows the relative error for each of those measurements. These results show that the measurement is reliable. Its accuracy will also be acceptable for a number of possible robotic applications.

4.5 Discussion

The measurement method has been formulated and its performances assessed. It was clearly demonstrated that this method provides reliable distance measurement information. Three important points were highlighted in the performance assessment. Firstly, the accuracy of the measure depends on precise estimation of the intrinsic and extrinsic camera parameters. If those are to be derived from calibration, this process must be closely monitored to avoid erroneous distance measurement at a later stage. Secondly, the error introduced by discrete pixel on the image is largely dependent, at a given distance, on the

camera location and orientation in space. One should carefully select these settings to optimize the measurement accuracy across the required range. Obviously the more pixels covering a given range the better the accuracy will be. A possible improvement would be to implement sub-pixel feature extraction. Finally, there are some camera postures for which no measurement will be possible. In cases were the focal axis is parallel to the plane defined by the X and Z axes of the robot frame, measurement will be totally impossible if the focal point also belongs to this plane. If the focal point is offset from this plane, measurement may only be possible across part of the image.

Despite the limitation enforced by the priori assumptions, this method will still prove useful to many mobile robotics applications. The effort to provide a good compromise between the required computational power and the richness of the information has remained the primary target. The ultimate goal would be the implementation of this method on small and relatively cheap hardware, to realistically envisage its use in large scale MRS.

4.6 Example Applications

This section presents two possible applications of the distance measurement algorithm. This is to demonstrate that from a simple distance measurement, one can implement a wide range of applications where a robot interacts with its environment.

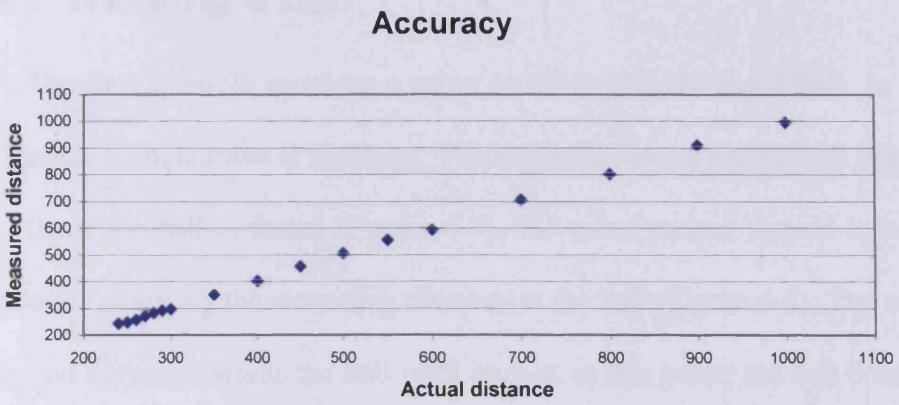


Figure 4-4. Accuracy of the Measurement in Practice

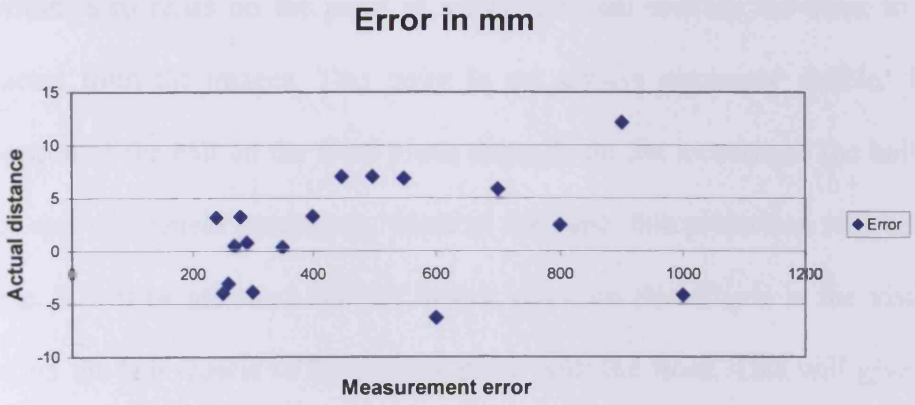


Figure 4-5. Error in Each of the Measurements

4.6.1 Tracking a Ball

The first example concerns a robot tracking and chasing a ball. In this application a single robot is involved. The robot first looks for the ball (Figure 4-6). Once the ball is found (Figure 4-7), the robot moves toward it while continually assessing the remaining distance to the ball (Figure 4-8). The robot carries on moving towards the ball until impact, at this point, the ball bounces off (Figure 4-9). The robot then starts looking for the ball again.

In this application, it is fair to assume that the floor is perfectly flat. Both intrinsic and extrinsic camera parameters are known. The measurement algorithm also relies on the point at which the ball touches the floor to be extracted from the images. This point is not always necessary visible. The projection of the ball on the focal plane depends on the location of the ball in space and the camera parameters. Most of the time, this projection will be an ellipse. It will be assumed that the lowest point on this ellipse is the visible point on the ball closest to the contact point with the floor. This will give an approximate distance to the ball, nevertheless good enough for the example application. The projection of the ball on the image is extracted using a simple colour differentiation algorithm, the assumption being that the ball is yellow and that no other element in the environment has the same colour.

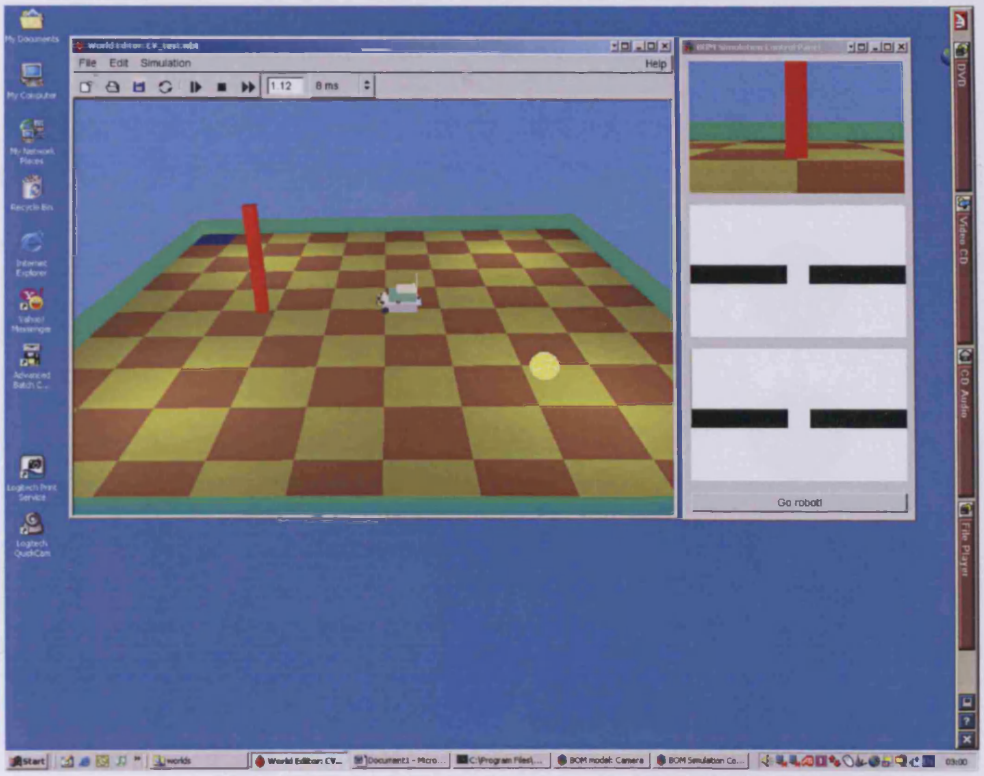


Figure 4-6. Robot Looking for the Ball

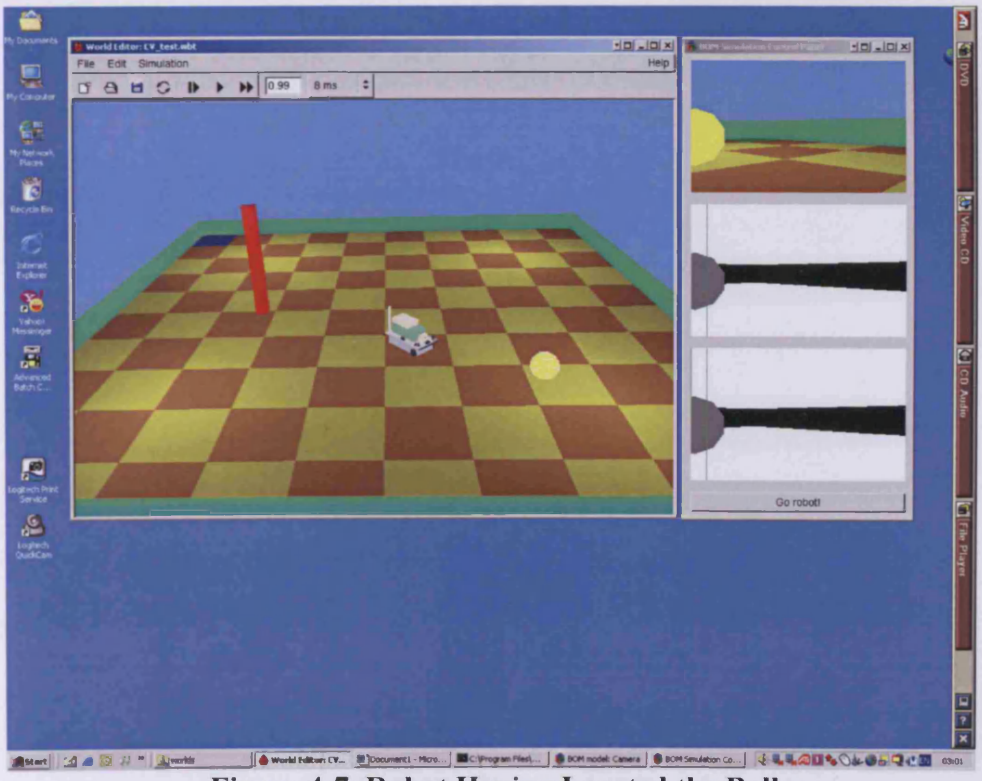


Figure 4-7. Robot Having Located the Ball

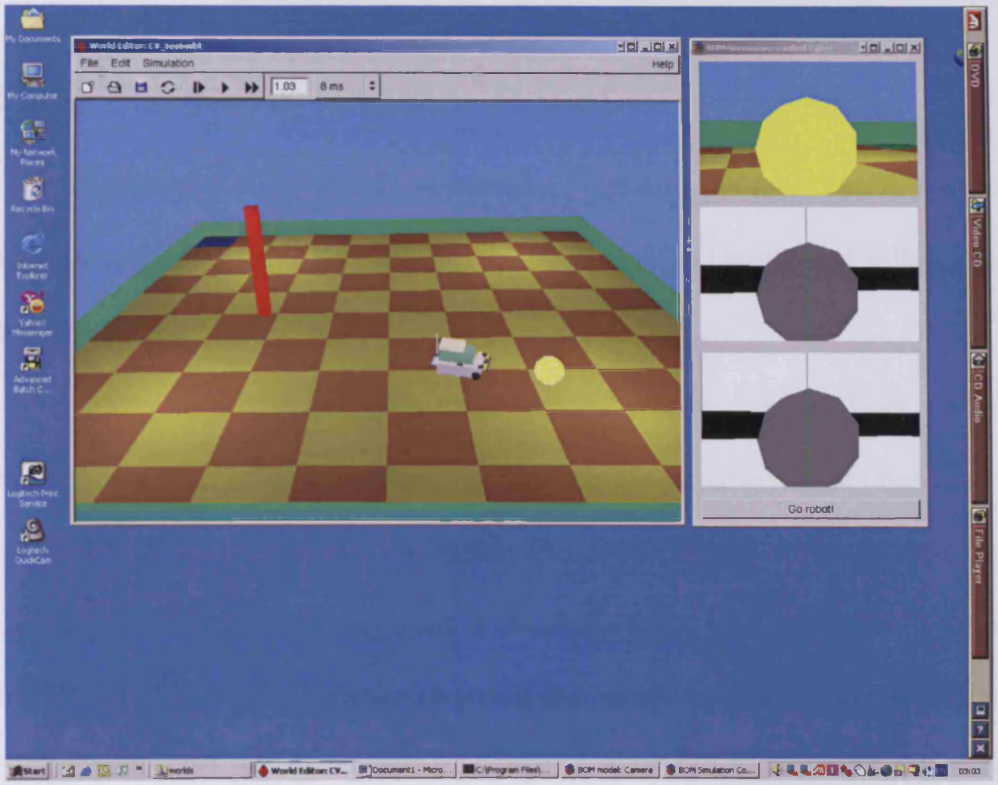


Figure 4-8. Robot Aiming for the Ball

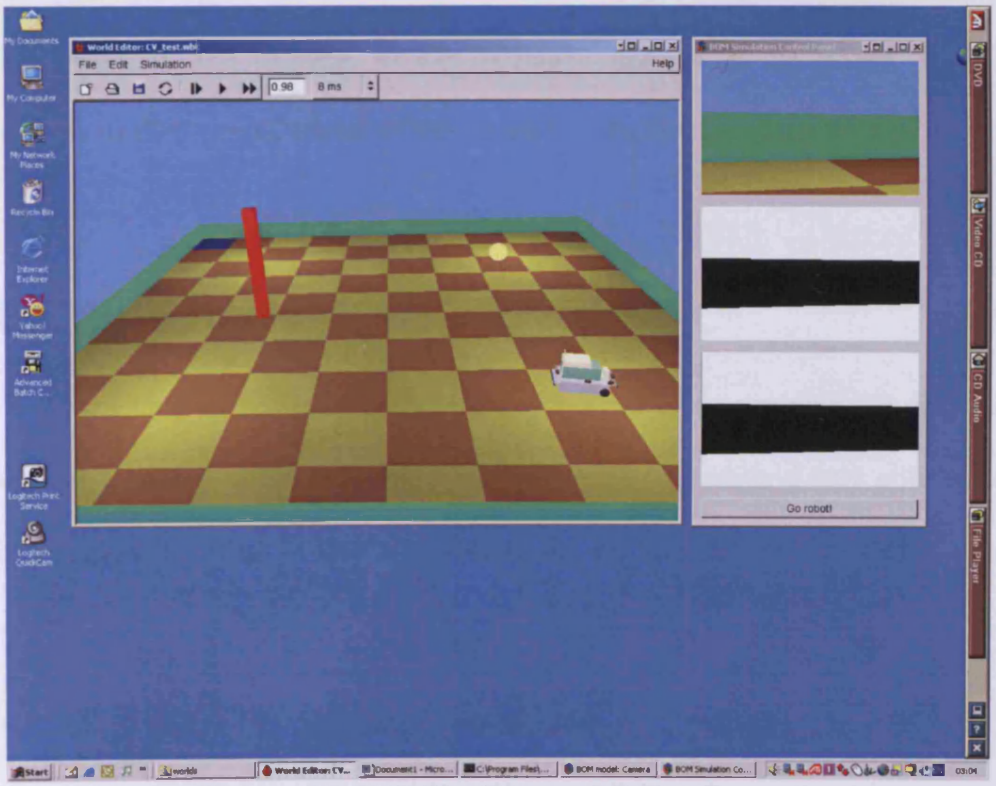


Figure 4-9. Ball Has Bounced Off on Impact

4.6.2 Following Another Robot

The second example involves a robot following another. The first robot wanders randomly around the environment. The second tracks and follows the first one (Figure 4-10).

Again, in this application, it is fair to assume that the floor of the environment is flat. Here the algorithm for measuring distances to objects at a known height off the ground, was used. The wandering robot was modified so that one if its building blocks bears a distinct colour. To second robot locates the first one by looking for and extracting this specific colour. It then computes the distance to the wandering robot. Using this information, it controls its speed in order to maintain a constant distance with the wandering robot. The follower also computes the location of the extracted feature on the image. It then controls its direction to maintain this feature in the middle of the field of view.

4.7 Summary

This chapter has presented a simple method of determining the distance between a mobile robot and objects in front of it. The method only utilise one camera and therefore is also adequate to be implemented on fleets of mobile robots.

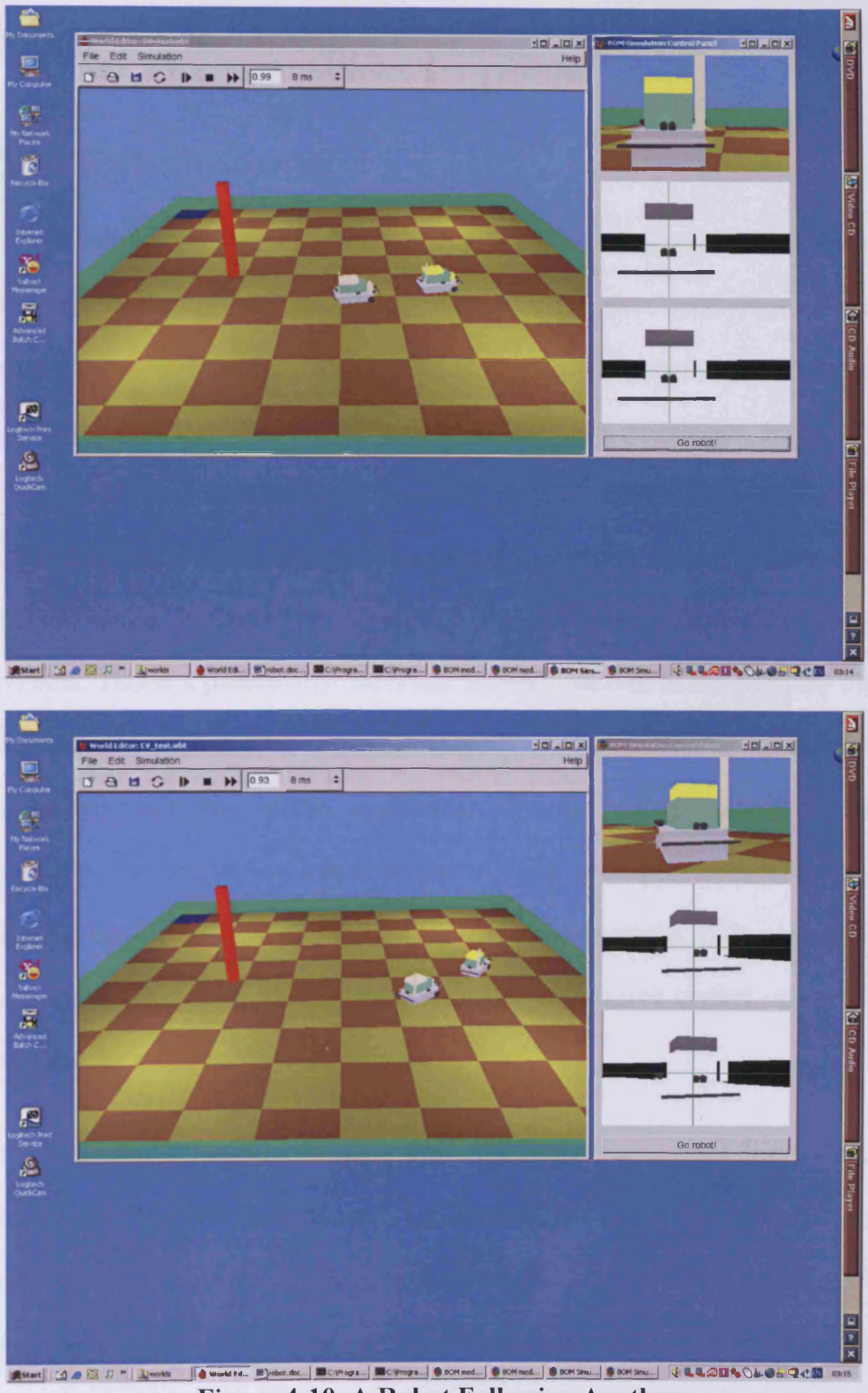


Figure 4-10. A Robot Following Another

Chapter 5 Conclusion and Future Work

5.1 Conclusions

The work presented in this thesis dealt with embodiment in Multi Robot Systems. In the last two decades, embodiment has developed from an idea behind the subsumption control architecture, to being formally defined. It has grown to be widely accepted throughout robotics and other fields. It has now emerged has a 'condition sine qua non' to any robotics system. Despite acknowledging its importance, the robotics community seems to have paid little attention to finding ways of best integrating embodiment to robotic systems. This is a particularly important shortfall because of the difficulty one faces in ensuring that good embodiment is achieved when developing an MRS. Robot systems at least include mechanical, electrical and electronic parts. The control of the robot is based on a number of sciences ranging from psychology and Artificial Intelligence to computing and vision systems. Only a carefully orchestrated integration of all those sciences will ensure correct interaction between the robot and its environment.

The first component of work presented in this thesis deals directly with this issue. The 'Re-embodiment' architecture has been put forward. 'Re-embodiment' provides a framework, through a number of Axioms, to implement Multi Robot Systems. In analogy with dualism, each system is considered to be made of a body and a soul. The boundary between the body

and the soul is arbitrarily set and not confined to the hardware/software distinction. The Axioms define the properties of both body and soul with respect to the environment and each other. It was demonstrated that, provided the Axioms are fulfilled, the resulting system will have a number of properties that improve its versatility. Firstly, the amount of work required to build a new system will be reduced. One may reuse a body with a soul to be tested or vice-versa. Managing large fleets of robots will be made easier. One may incarnate a single soul into several bodies; one may force a robot to clone its soul into other bodies. One may deploy a soul into a fleet of bodies even when some of the bodies are not within range of the starting point of the soul. Implementing co-evolution and mutation into an MRS will be much facilitated. One may easily implement triggers in a soul that activate mutation or cloning of several souls followed by re-embodiment. Multiple users working on largely different issues with limited numbers of units can operate at virtually the same time. Thanks to the Re-embodiment flexibility, the use and efficiency of a platform can be optimised to allow improved research and development. The possibilities do not stop here. One may implement an architecture where a soul can incarnate different types of physical body or even virtual bodies evolving in virtual environments.

Although no metric yet exists for embodiment, it has been clearly demonstrated that it is not a discrete state. As per the definition of embodiment there exist perturbatory channels between the system and the environment. The degree of perturbation on those channels may vary, mainly based on the

sensory and motory ability of the robot. A common mistake across MRS research is that lower sensory and motory capabilities, compared to single robot systems, are deemed acceptable because of the increased number of units. This however contradicts the basic principle of embodiment. On the other end, it is understandable that budgets enforce limits on the costs of sensors and actuators for each unit.

To directly address this issue an information rich sensory input that can be cheaply implemented was developed. The method allows for 'Robot-to-Object' distance measurement from a single image frame. It is assumed that the intrinsic and extrinsic camera projection parameters can be determined through a one-off calibration. It is further assumed that the height of the target can be determined by supposing that the floor of the environment in which the robot evolves is flat and that the object is either at a known height from the floor or in contact with it. Based on this, it is possible to extract distances between objects on the image and the robot. The measurement system was implemented using inexpensive 'Web-cams'. The accuracy of the measurement system was studied firstly through theoretical analysis and secondly through a series of experiments. Finally, two example applications were presented in order to demonstrate the large range of applications that can benefit from such distance measurement method. In the first one, a robot tracks a ball, in the second a robot follows another.

5.2 Summary of Contributions

This section runs through the contributions made by the work presented in this thesis. Those contributions relate to a common subject: The importance of embodiment in robotics. The focus is on embodiment in Multi Robot Systems.

- Provide an architecture for MRS development platforms that eases implementation of embodied systems.
- Provide an architecture for MRS development platforms that is flexible and scalable to large numbers of units.
- Provide an architecture for MRS development platforms that allows for several users to work at virtually the same time on largely different systems.
- Provide an information rich sensory input, that allows for better embodiment.
- Provide a sensory input that is relatively cheap to implement, to allow realistic use on large numbers of units.

5.3 Suggestions for Future Work

Since the late nineteen seventies, robotics, as a research field, has matured extensively. It has emerged as a meeting point for many sciences. Research is broad and moving forward on many fronts. Multi Robot Systems and cooperation have been at the forefront of this effort. Despite many breakthrough and advances, robotics as a science is still far from achieving the target set by the imagination. The old dream, of being able to understand and mimic intelligence, remains beyond grasp.

5.3.1 Development Platform for Large Scale MRS

One of the main breakthroughs at the end in the mid nineteen eighties was the emergence of embodiment. Despite its acceptance across the field since then, there remain huge gaps in its standardization in robotic systems. One of these gaps is the repeated effort and time spent by researchers, even in the same lab, to implement an embodied development platform. This is especially damaging to advances in the field because research only really begins once such a platform is operational. Furthermore, in robotics, findings are only seen as valid once tested in the real world on such a platform. Too often most of the research effort is spent on the implementation of such platform. This is even more damaging when a platform has no flexibility and can only be used for one particular test, thus forcing researchers to implement another all over again before further advances can be attempted. Part of

robotics is about building the robot themselves but one must ensure that researching new theories and making those robots perform remains just as important. The Re-embodiment architecture was shown to provide such advantages, but more can be done. Although the Re-embodiment implementations described in this thesis have shown to be scaleable, more tools would probably be needed to manage very large fleets:

Firstly, a tool to record and monitor which soul has incarnated which body would be required. All available souls should be clearly listed as ready of use; all different bodies should also be classified with clear descriptions of capabilities should they be real robots or simulated agents evolving in virtual environments. All bodies should also be monitored allowing status checking at a glance, in order to minimize the effort required to maintain the fleet operational. The ability to detect faulty units or those out of power should be implemented to maximise the fleet availability.

Secondly each unit in a very large fleet should have the ability to manage its own power consumption and requirement. With growing numbers of units there will be a point where it would be practically impossible to have to manually recharge units to keep the system running. It is relatively easy to envisage the implementation of charging stations were units could come to refill their power supply. In very widespread systems, where it is impractical to travel back to charging stations, one could even implement ways of conveying power from one unit to another across the fleet.

Finally, and above all, the main emphasis should be the design of a multi user interface. To fully exploit the advantage of easy embodiment on large

MRS, each user should be able to work on their respective projects at the same time with a unique platform. One could develop a system where researchers request the use of bodies from a shared pool. They could also share their work by making available to others souls or bodies they have developed. These improvements are seen as important and urgently needed, especially considering recent developments with relatively large robot fleets. The Centibots project at SRI boasts 100 units **[Konolige *et al.*, 2002]**. More recently a student at MIT built 112 robots **[Miranda, 2004]**.

5.3.2 Improving Cheap yet Information Rich Sensing

The foremost step towards standardization of Embodiment in robotic systems was the appearance of its formal definition **[Quick and Dautenhahn, 1999]**. This formal qualification made it clear that embodiment is not a discrete state and that different degrees of embodiment may occur depending on the motory and sensory abilities of the robot. Obviously one should try to maximise the embodiment of a robot. In MRS, cost restrictions often limit the computational power and sensor complexity of each unit.

The mono vision distance measurement method put forward in this thesis addresses both these problems. Although the advantage of this measurement method is clear, the potential for further improvement remains high.

Firstly, the required calibration to determine the extrinsic and intrinsic parameters of the camera is somewhat tedious In fact, it would quickly become impractical with large MRS. Furthermore, any physical modification to the

system, such a disassembly and reassembly would require recalibration. There exist numerous ways of calibrating cameras. Recently, calibration methods using references from the scene and known motions have been developed. Should such methods provide sufficiently accurate results, they would be a great improvement as each robot could self-calibrate its camera.

Secondly, the measurement method relies on the ability to detect a point or an edge on an object. To ensure that implementation in robots with less computational power, such as the Sheepbots, is possible, one must ensure that image analysis algorithms to extract features, simple enough to be realised on small programmable controllers, are available.

Thirdly, the measurement method, as presented here, relies on an environment with a planar floor. Although this is a perfectly valid assumption to be made, in a number of applications it is somehow limiting. There would be an easy way for the robot to at least check whether this assumption can be made. Suppose that the robot can detect a feature in contact with the floor on an object and assess the distance to this object. Assuming that the robot can measure distances travelled through odometry, it can then move by a known distance. This distance should be known precisely enough since it is only measured over a short time. The distance between the robot and the target can then be re-assessed. If the floor really is flat, the distance travelled by the robot should match the difference between the distances to object before and after the motion. In the case of a non planar floor, these distances will not match. Through this simple method the robot should be able to continually check the

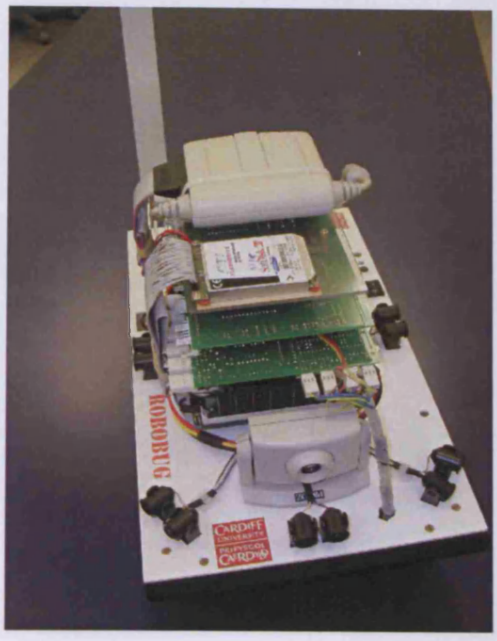
floor flatness assumption, allowing for measurements to be discarded when non planar floor are detected.

Finally, a plethora of new algorithms and applications can be developed by exploiting the robot-to-object distance measurement. One can imagine applications such as group formation, where distance and orientation between robots need to be known. Assume that each robot in an MRS is fitted with a camera providing 360 degree field of view. Each robot should be able to compute the position of other robots around it by measuring angles and distances to each other unit. This could be used for applications such as flocking or multi target tracking.

5.3.3 Standardizing Cooperation

One of the next important steps for the future of MRS will be the standardization of cooperation. Many claim to have achieved cooperation with two or more robots. However, the definition of what cooperation really is, the setting of the experiment or application and the results greatly differ. A unified definition of cooperation will be the first step towards qualifying it. Then based on this definition one may start considering quantifying it.

Appendix A - BOM Robots' Specifications



Name: Robobug[2]

PC board: AR-B1474 (Half Size 486DX4/DX2 CPU Card).

CPU: Intel 486DX2 33MhZ.

Motor interface board: Custom made motor interface board.

Sensor board: Custom made motor interface board.

Wireless network card: ORiNOCO Silver PCMCIA 802.11B Card

PCMCIA to PC/104 adapter: JUMPtec PC/104-PCMCIA-1 Adapter

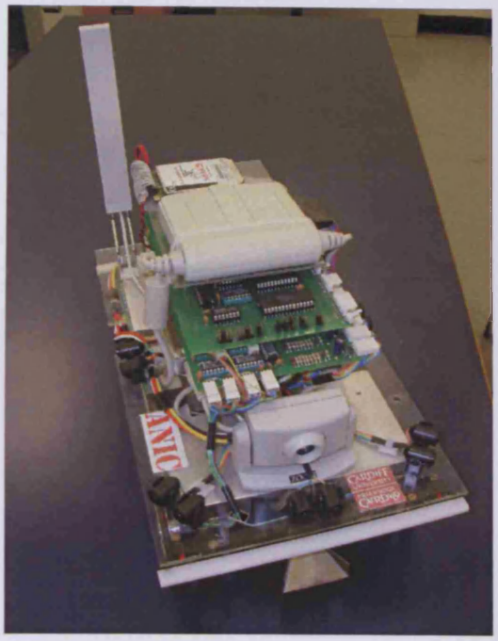
Storage: 4MB Flash Drive

Sensors: four bumper switches, six ultrasound sensors.

Camera: ZoomCam PPC.

Operating system: Small print, custom operating system based on Linux.

[2] Robobug was named so because it was the first of the BOM robots on which the custom operating system and the Re-Embodiment architecture were implemented and debugged. It is not related to the Mekatronix walking machine commercial kit of the same name (http://www.mekatronix.com/detailed/robobug.htm).



Name: Anic[3]

PC board: PIA-460 (half-sized Pentium PCI/ISA CPU Card with VGA)

CPU: Intel Pentium I 66MHz.

Motor interface board: Custom made motor interface board.

Sensor board: Custom made motor interface board.

Wireless network card: ORiNOCO Silver PCMCIA 802.11B Card

PCMCIA to PC/104 adapter: JUMPtec PC/104-PCMCIA-1 Adapter

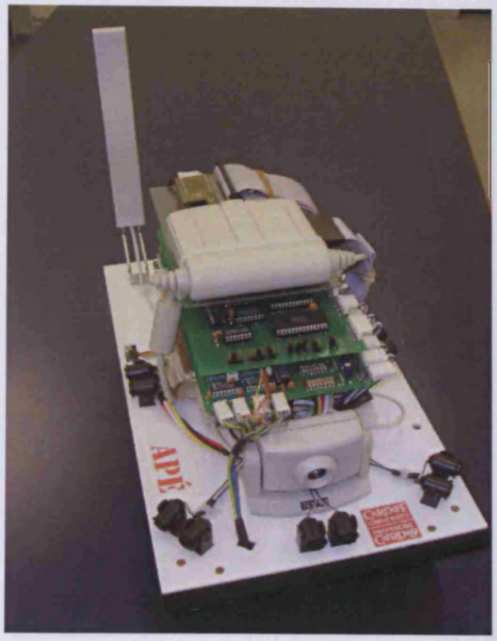
Storage: 4MB Flash Drive

Sensors: four bumper switches, six ultrasound sensors.

Camera: ZoomCam PPC.

Operating system: Small print, custom operating system based on Linux.

[3] Botanic: adjective, from French *botanique*, from Greek *botanikos* of herbs, from *botane* pasture, herb, from *boskein* to feed, graze; probably akin to Lithuanian *guotas* flock. 1: of or relating to plants or botany. 2: derived from plants



Name: Apé[4]

PC board: PIA-460 (half-sized Pentium PCI/ISA CPU Card with VGA)

CPU: Intel Pentium I 66MHz.

Motor interface board: Custom made motor interface board.

Sensor board: Custom made motor interface board.

Wireless network card: ORiNOCO Silver PCMCIA 802.11B Card

PCMCIA to PC/104 adapter: JUMPtec PC/104-PCMCIA-1 Adapter

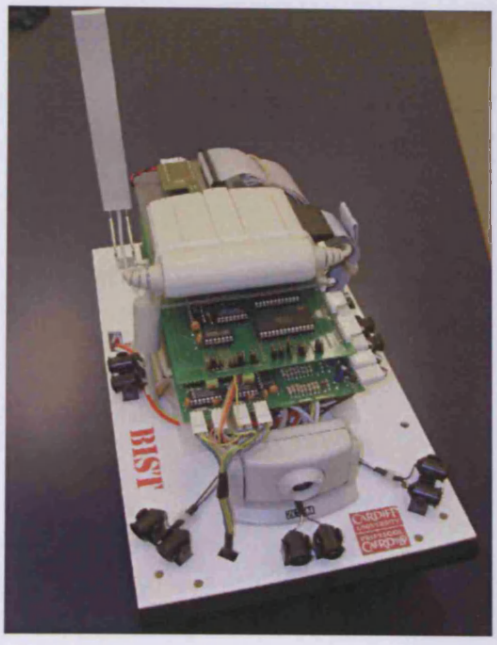
Storage: 4MB Flash Drive

Sensors: four bumper switches, six ultrasound sensors.

Camera: ZoomCam PPC.

Operating system: Small print, custom operating system based on Linux.

[4] Apéro: noun, diminutive of *apéritif*, from Medieval Latin *aperitivus*, irregular from Latin *aperire*, an alcoholic drink taken before a meal as an appetiser.



Name: Bist[5]

PC board: PIA-460 (half-sized Pentium PCI/ISA CPU Card with VGA)

CPU: Intel Pentium I 66MHz.

Motor interface board: Custom made motor interface board.

Sensor board: Custom made motor interface board.

Wireless network card: ORiNOCO Silver PCMCIA 802.11B Card

PCMCIA to PC/104 adapter: JUMPtec PC/104-PCMCIA-1 Adapter

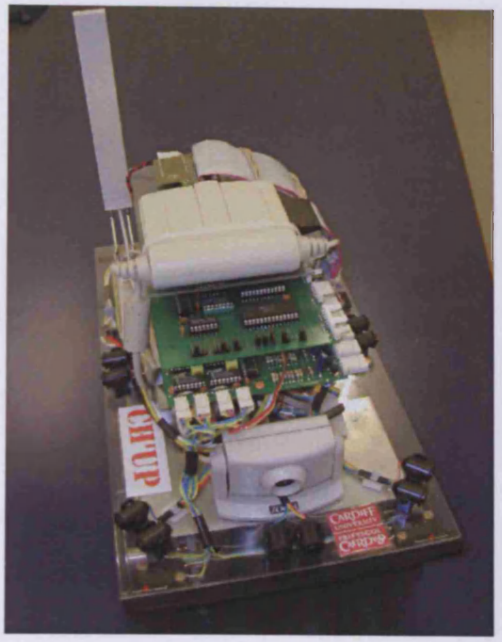
Storage: 4MB Flash Drive

Sensors: four bumper switches, six ultrasound sensors.

Camera: ZoomCam PPC.

Operating system: Small print, custom operating system based on Linux.

[5] Bistro: noun, from French, 1: a small or unpretentious restaurant, 2: a small bar or tavern.



Name: Ch'up[6]

PC board: PIA-460 (half-sized Pentium PCI/ISA CPU Card with VGA)

CPU: Intel Pentium I 66MHz.

Motor interface board: Custom made motor interface board.

Sensor board: Custom made motor interface board.

Wireless network card: ORiNOCO Silver PCMCIA 802.11B Card

PCMCIA to PC/104 adapter: JUMPtec PC/104-PCMCIA-1 Adapter

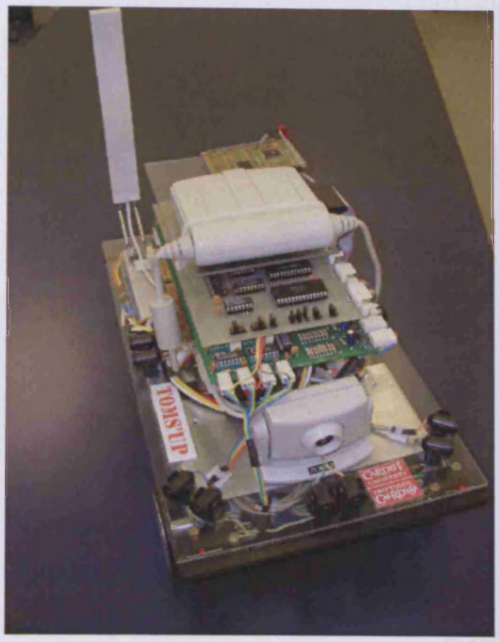
Storage: 4MB Flash Drive

Sensors: four bumper switches, six ultrasound sensors.

Camera: ZoomCam PPC.

Operating system: Small print, custom operating system based on Linux.

[6] Botch'up: transitive verb, from Middle English *bocchen*, 1: to foul up hopelessly, 2: to put together in a makeshift way.



Name: Toms'up[7]

PC board: PIA-460 (half-sized Pentium PCI/ISA CPU Card with VGA)

CPU: Intel Pentium I 66MHz.

Motor interface board: Custom made motor interface board.

Sensor board: Custom made motor interface board.

Wireless network card: ORiNOCO Silver PCMCIA 802.11B Card

PCMCIA to PC/104 adapter: JUMPtec PC/104-PCMCIA-1 Adapter

Storage: 4MB Flash Drive

Sensors: four bumper switches, six ultrasound sensors.

Camera: ZoomCam PPC.

Operating system: Small print, custom operating system based on Linux.

[7] Bottoms'up: expression, from English, drinking the whole glass of something all at once (usually beer).

Appendix B -
Source Code of the 'linuxrc' Script

```
#!/bin/sh


#This script tries to retrieve a root image over the wireless network
# link. The script first searches for an image in a directory common
# to all robots in the fleet, if nothing was found there, the script
# search for an image in a directory specific to the robot.
# The script try to be as robust as possible and check for:
#    -No file in directory
#    -File not readable
#    -File not a ext2 fs image
#    -Problem when transferring file (often size mismatch)
#      between image and ramdisk ( image> ramdisk)
#    -Several files in the directory no read after
#      successful transfer of an image


#Change log
#
#Version 1.0.1
# +First stable release
# +This script tries to retrieve a root image over the wireless network

#  link. The script first searches for an image in a directory common
#  to all robots in the fleet, if nothing was found there, the script
```

```
#  search for an image in a directory specific to the robot.
# +If server unreachable or no image do download the default root
#  image is used
# +The script try to be as robust as possible when downloading image
#  and check for:
#    -No file in directory
#    -File not readable
#    -File not a ext2 fs image
#    -Problem when transferring file (often size mismatch)
#      between image and ramdisk ( image> ramdisk)
#    -Several files in the directory, no read after
#      successful transfer of an image
# +If a valid root image is downloaded, log generated while the default
#  root image is mounted as root, are transferred to the newly mounted
#  root image before the default root image is unmounted.
#
#
#Version 1.0.2
# +Added robot specification information as part of the image
# +If a valid root image is downloaded the robot specific information
#  are transferred onto the downloaded root image  before the default
#  image is unmounted
#
#
#Version 1.0.3
# +The PCMCIA modules are unloaded if an image was successfully
```

```
#  retrieved from the boot server so that problems are avoided if
#  the PCMCIA modules (from the initrd image) and the cardmgr (from
#  the downloaded image) don't match.
# +Improved message displayed on console during image download
#
#
#Version 1.0.4
# +Updated image so that it includes the utilities required to update
#  the initrd image on the robot. It avoid having to load a utility
#  image when the initrd image is to be updated on the robot
#
#Version 1.0.5
# +Updated script and image to work with 2.2.19-7.0.1 Kernel
# +Use of a variable for kernel version
# +Had to mount nfs partition using nolock, as I was getting
#  error messages, but don't know why??
# +Fix problem with time, use /usr/bin/time as time is now also a bash built-in command
#
#Version 1.0.6
# +Reverted back to work with kernel 2.2.19-6.2.1
# +Use BOM major number 251
# +Mounting with lock again
```

```
#*********************************************************************
#This part of the script will set a few parameters

  #ROBOT_NAME=
  ROBOT_SPECIFICINFO_FILE=/etc/bomrobotinfo
  ROBOT_NFS_MOUNTPOINT=/mnt/temp/
  ROBOT_RAMDISK=/dev/ram1
  ROBOT_RAMDISK_MAJMINNUMBER=0x100
  ROBOT_OLDLOGTRANSFER_MOUTPOINT=/mnt/temp/
  BOOTSERVER_IP=10.0.0.10
  BOOTSERVER_BOMFLEET_ROOT=/home/bomfleet/
  BOOTSERVER_BOMFLEET_COMMONDIR=all
  INITRD_VER=1.0.6
  KERNEL_VER=2.2.19-6.2.1


#*********************************************************************
#This part of the script will set the PATH,
# mount a proc filesystem and start the system and
# kernel loggers.

  # Set the path
  PATH=/bin:/sbin:/usr/bin:/usr/sbin
  export PATH

  #Mount proc fs (Cardmgr requires proc to be mounted)
```

```
echo "Mounting proc filesystem"; mount -nt proc /proc /proc
sleep 1

#Start syslogd and klogd
echo "Starting System logger"
syslogd
klogd

#Echo and log initrd root image version
echo "initrd root fs image version $INITRD_VER"
initlog -s "initrd root fs image version $INITRD_VER"



#*********************************************************************
#This part of the script read robot specific parameter from a file
# and log/display robot's name

  #Retrieve robot specific info
  if [  -f /etc/bomrobotinfo ] ; then
     #Get specific BOM robot information from file
     . /etc/bomrobotinfo
     echo "Robot specific information retrieved"
     initlog -s "Robot specific information retrieved"

  else
```

```
    #There is a problem, robot specific information are missing
    #use some default values, and log problem
    ROBOT_HOSTNAME=problem.bom
    echo "Could not find robot specific information, using default"
    initlog -s "Could not find robot specific information, using default"
  fi

  #Echo and log robot name
  echo "My name is $ROBOT_HOSTNAME.bom"
  initlog -s "My name is $ROBOT_HOSTNAME.bom"



#*********************************************************************
#This part of the script will load the BOM device module
# and reset the motor, to stop them if they start turning
# at power up.

  #Insert BOM device module
  echo "Loading BOM device driver and resetting motors"
  initlog -s "Loading BOM device driver and resetting motors"
  /sbin/insmod "/lib/modules/$KERNEL_VER/misc/motor2.o" bom_major=251
  echo _IAx > /dev/mot
```

```
#*********************************************************************
#This part of the script will load required modules
# for the PCMCIA Wireless network card and start the
# card manager.

  # PCMCIA configuration
  PCIC="i82365"
  PCIC_OPTS=""
  CORE_OPTS=""
  CARDMGR_OPTS=""

  #Output message
  echo "Starting PCMCIA services"

  #Insert the required modules
  /sbin/insmod "/lib/modules/$KERNEL_VER/pcmcia/pcmcia_core.o" $CORE_OPTS
  sleep 1
  /sbin/insmod "/lib/modules/$KERNEL_VER/pcmcia/$PCIC.o" $PCIC_OPTS
  sleep 1
  /sbin/insmod "/lib/modules/$KERNEL_VER/pcmcia/ds.o"
  sleep 1

  #Configure the card
     #Manually, can't get it to work (it can't recognise eth0!!)
     #/sbin/insmod /lib/modules/2.2.14-5.0/pcmcia/wavelan2_cs.o port_type=3
     #/sbin/ifconfig eth0 up 10.0.0.4 255.255.255.0 10.0.0.255
```

```
    #Using cardmgr, it works fine, so what the h##l
    /sbin/cardmgr $CARDMGR_OPTS
    sleep 3


#*****************************************************************
#This part of the script will retrieve the filesystem
# from isl-1.bom:/home/bomfleet and copy it into a ramdisk
# using nfs. Check are made for non-valid files, multi
# files and non-existent files. If no file is present on
# the boot server, the default fs (already loaded in ram0) is
# set as the main root filesystem and init can be started
# from it.


  #Insert module required for nfs
  /sbin/insmod "/lib/modules/$KERNEL_VER/misc/sunrpc.o"
  /sbin/insmod "/lib/modules/$KERNEL_VER/fs/lockd.o"
  /sbin/insmod "/lib/modules/$KERNEL_VER/fs/nfs.o"

  #temp stuff
  #sleep 1
  #cat /proc/modules
  #sleep 10
```

```
GOTIMAGE=false
#Try to mount nfs
if mount -n "$BOOTSERVER_IP:$BOOTSERVER_BOMFLEET_ROOT" $ROBOT_NFS_MOUNTPOINT ; then #-o nolock

   #Check if an image file is present in common directory then in the robot specific directory
   echo "File system image search on the boot server"
   GOTIMAGE=false
   for SEARCHDIR in "$ROBOT_NFS_MOUNTPOINT$BOOTSERVER_BOMFLEET_COMMONDIR/" "$ROBOT_NFS_MOUNTPOINT$ROBOT_HOSTNAME/"; do
      echo "   Searching in $SEARCHDIR"
      LIST="`ls $SEARCHDIR`";
      #echo "     file(s) $LIST was/were found on boot server in $SEARCHDIR directory"
      initlog -s "file(s) $LIST was/were found on boot server in $SEARCHDIR directory"
      for IMAGE in $LIST; do
         if [  "$GOTIMAGE" = "false" ]; then
            if [  -n "$IMAGE" ]; then
               if [  -r "$SEARCHDIR$IMAGE" ]; then
               #Check if file is a valid ext2 fs image
                  RESULT=`fsck.ext2 -p "$SEARCHDIR$IMAGE" 2>&1`
                  if echo "$RESULT" | grep "clean" >/dev/null; then
                     initlog -s "$RESULT"
                     #Retrieve filesystem image and copy into ramdisk
                     echo "      Image file $IMAGE being retrieved from network..."
                     initlog -s "Image file $IMAGE being retrieved from network"

                     if RESULT=`/usr/bin/time dd if="$SEARCHDIR$IMAGE" of=$ROBOT_RAMDISK bs=1k 2>&1`; then
```

```
                echo "      ...Image downloaded successfully"
                initlog -s "$RESULT"
                GOTIMAGE=true
             else
                echo "      ...Download aborted!! check log for details"
                initlog -s "$RESULT"
                echo "      Warning, image file $IMAGE not retrieved, may be size mismatch!"
                initlog -s "Warning, image file $IMAGE not retrieved, may be size mismatch!"
             fi
          else
             initlog -s "$RESULT"
             echo "      Warning, image file $IMAGE is not a valid ext2 fs"
             initlog -s "Warning, image file $IMAGE is not a valid ext2 fs"
          fi
       else
          echo "      Warning, image file $IMAGE is not readable"
          initlog -s "Warning, image file $IMAGE is not readable"
       fi
    else
       echo "   Warning, their is no file in directory!"
       initlog -s "Warning, their is no file in directory!"
    fi
 else
    echo "      Warning, a root image was found but there are other file here"
    initlog -s "Warning, a root image was found but there are other file here"
    break
```

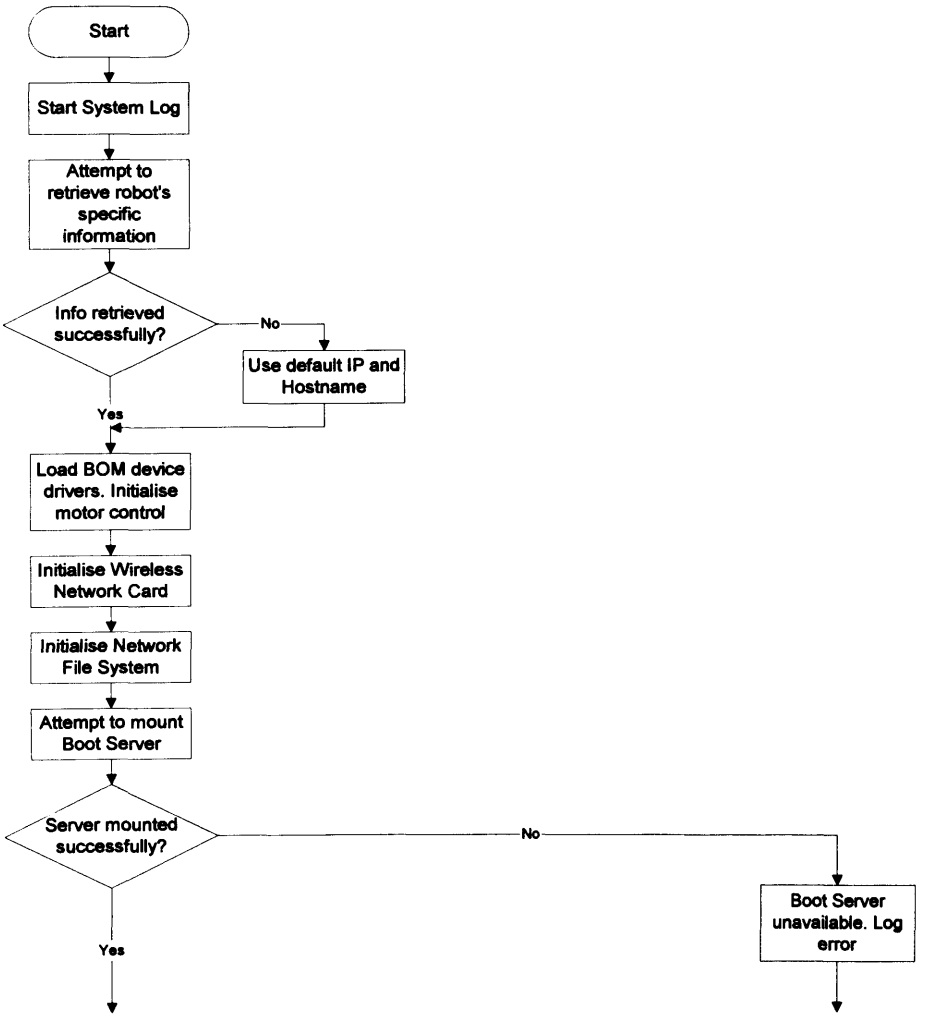
```
        fi
      done
   done
   #If no image was downloaded mount default rescue system as root
   if [  "$GOTIMAGE" = "false" ]; then
      echo "   Warning, no image found on server, starting default root image"
      initlog -s  "Warning, no image found on server, starting default root image"
      echo $ROBOT_RAMDISK_MAJMINNUMBER > /proc/sys/kernel/real-root-dev
   fi

   #Unmount network file system
   umount -n $ROBOT_NFS_MOUNTPOINT

else
   #Server unreachable for some reason, did not try to retrieve image
   # mount the default rescue system as root
   echo "Could not mount nfs, not trying to retrieve fs image"
   initlog -s "Could not mount nfs, not trying to retrieve fs image"
   GOTIMAGE=false
   echo $ROBOT_RAMDISK_MAJMINNUMBER > /proc/sys/kernel/real-root-dev

fi

#Remove nfs modules
rmmod nfs
rmmod lockd
```

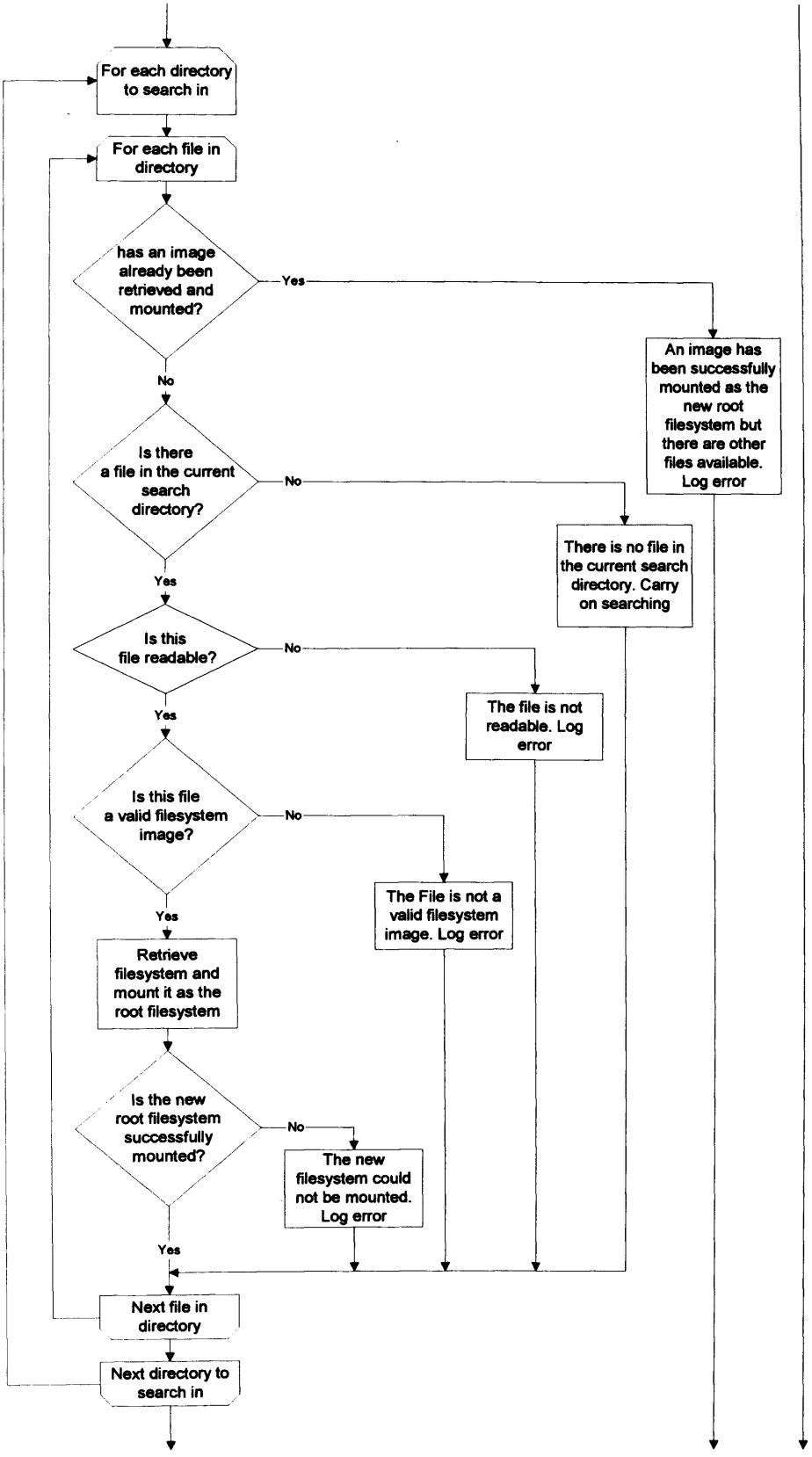
```
      rmmod sunrpc


#*********************************************************************
#This part of the script will perform some cleanup
# before the temporary root filesystem can be unmounted, remaining
# process have to be killed and the logfiles copied to the new fs
# for future reference. If the temporary root fs is to become
# the main root fs, those process can carry on


      if [  "$GOTIMAGE" = "false" ]; then

              initlog -s "No fs image retrieved, process lunched by linuxrc can carry on"
              sleep 3

      else

              #kill cardmgr (and wait a bit)
              PID=`cat /var/run/cardmgr.pid`
              echo "The PID of cardmgr is $PID, killing cardmgr"
              initlog -s "The PID of cardmgr is $PID, killing cardmgr"
              kill $PID
              sleep 4

              #Remove pcmcia modules
              echo "Unloading pcmcia modules"
              /sbin/rmmod ds
              sleep 1
              /sbin/rmmod i82365
              sleep 1
              /sbin/rmmod pcmcia_core
              sleep 3

              #Kill syslogd and klogd
              PID=`cat /var/run/syslogd.pid`
              echo "The PID of syslogd is $PID, killing syslogd"
              initlog -s "The PID of syslogd is $PID, killing syslogd"
```

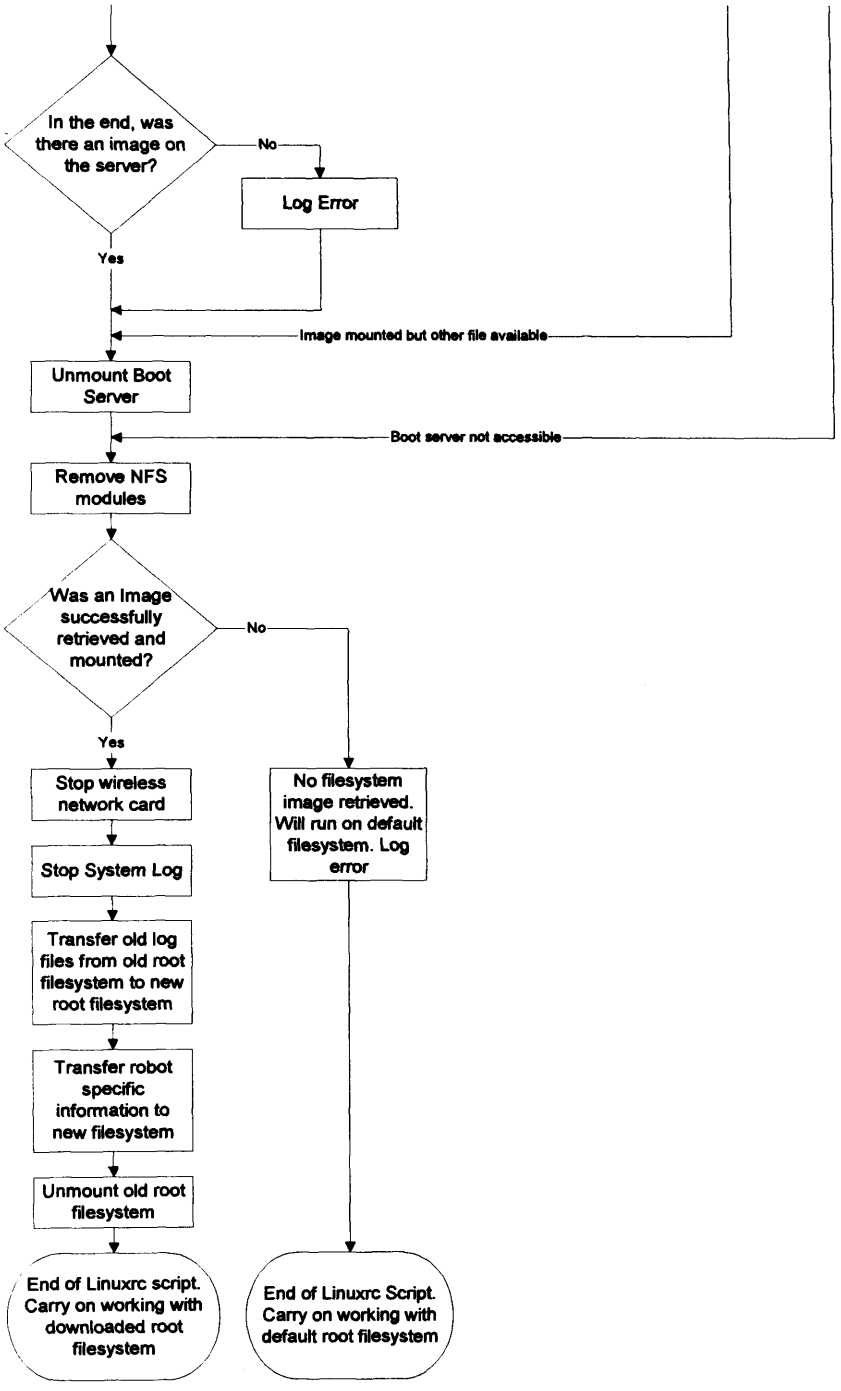
```
        kill $PID
        PID=`cat /var/run/klogd.pid`
        echo "The PID of klogd is $PID, killing klogd"
        initlog -s "The PID of klogd is $PID, killing klogd"
        kill $PID
        sleep 1

        #Transfer some info to new root fs (Cannot be logged)
        ##Remount old root
        mount $ROBOT_RAMDISK  $ROBOT_OLDLOGTRANSFER_MOUTPOINT
        ##Transfer old log file into the new fs
        echo "Transferring old log file on new filesystem"
        #initlog -s "Transferring old log file on new filesystem"
        cat < /var/log/messages > /mnt/temp/var/log/messages
        ##Transfer robot specific information into the new fs
        echo "Transferring robot specific information file on new filesystem"
        #initlog -s "Transferring robot specific information file on new filesystem"
        cat < "$ROBOT_SPECIFICINFO_FILE" > "/mnt/temp/$ROBOT_SPECIFICINFO_FILE"
        ##Unmount old root
        umount $ROBOT_RAMDISK
        sleep 1

        #temporary check for remaining process
        #ps -A
        #sleep 50

        #unmount the /proc filesystem
        echo "Unmounting proc filesystem"
        initlog -s "Unmounting proc filesystem"
        umount -n /proc
        sleep 1

fi
```

Appendix C -

'linuxrc' Script Flowchart




For each directory to search in
For each file in directory
has an image already been retrieved and mounted?
Yes
An image has been successfully mounted as the new root filesystem but there are other files available. Log error
No
Is there a file in the current search directory?
No
There is no file in the current search directory. Carry on searching
Yes
Is this file readable?
No
The file is not readable. Log error
Yes
Is this file a valid filesystem image?
No
The File is not a valid filesystem image. Log error
Yes
Retrieve filesystem and mount it as the root filesystem
Is the new root filesystem successfully mounted?
No
The new filesystem could not be mounted. Log error
Yes
Next file in directory
Next directory to search in


In the end, was there an image on the server?
No
Log Error
Yes
Image mounted but other file available
Unmount Boot Server
Boot server not accessible
Remove NFS modules
Was an Image successfully retrieved and mounted?
No
Yes
Stop wireless network card
Stop System Log
Transfer old log files from old root filesystem to new root filesystem
Transfer robot specific information to new filesystem
Unmount old root filesystem
End of Linuxrc script. Carry on working with downloaded root filesystem
No filesystem image retrieved. Will run on default filesystem. Log error
End of Linuxrc Script. Carry on working with default root filesystem

Appendix D - Experimental Assessment of the Re-Embodiment Implementation on the BOM Robots.

Aims

The aim of this series of experiments was to assess the performances of the development platform, implemented on the BOM robot along the Re-Embodiment principles.

The objective was to time the boot process in different conditions to get a general idea of the amount of time required to get an up and running system.

Experimental Condition

The timing of the boot process was measured manually using a conventional time watch from the time when the power switch is flicked on, to the time when the system is running (login prompt waiting). Because the robots are not fitted with a display device, a small application was developed to produce a double beep when the system is running. It is important to note that because of the nature of the boot process software timing could not be implemented.

The time required to download an eventual file system was also recorded. This was timed using software to achieved better precision.

Finally, it was observed during use that the battery voltage and the power supply greatly affects the wireless network card. For instance placing the wireless network card at different positions in the PC/104 stack would result in great differences in signal quality and overall performance. It was observed that better results were achieved when the card was as close as possible to the CPU board and the Power distribution board. Consequently, on the robot the wireless network card is placed between the CPU board and the power supply board in the stack. The Wireless network card also seems to be the first element of the system to be affected when the battery starts to be discharged. It was observed that when the battery voltage drops below approximately 11.5V the wireless card ceases to function properly.

To insure that experimental results are not distorted, the battery voltage was monitored during experiments.

Measurements

In the first series of tests, the boot process was timed in the following conditions. The boot server was running and accessible but no image was available. The boot process for each robot was first timed individually. Then, using the same conditions, the robots where timed when started simultaneously. Each measurement was repeated ten times.

In the second series of tests, a four Megabytes image was available for each robot to download. Firstly, the boot process for each robot was timed individually. Then, the boot process for each robot was timed when they were all started simultaneously (and supposedly attempting to download their respective image simultaneously). Each measurement was repeated ten times.

In the third and fourth series of test an eight and sixteen megabytes image were available for each robot to download respectively. The same measurements were made as previously.

Variation in Raw Data

There were significant variations in the measurements obtained. This could be due to a number of factors. First, the manual timing is prone to error due to variations in human reaction time.

However, there were also variations in the download timings. Linux is a multi-tasks and multi-thread operating system. Several processes appear to be running at the same time on a single processor. Whereas, in fact, the kernel scheduler distributes the processor resources to different processes and interrupt handlers. The amount of processor time allocated to the processes affecting the download on both the server and the robot may vary. This may result in variation in the overall download time. Interference to the RF signal could also affect the performances. Finally, when several robots attempt to

download filesystems, the wireless network bandwidth may affect the download time.

Results

Unexpectedly one of the Pentium robots (Bist) appears to be consistently slower than the other four. This difference occurs only in the overall boot up time not in the download time. This could not be explained. Consequently the data taken from Bist was discarded in the average time calculation.

Results show that the average boot-up times for the 486 and the Pentium when there is no image do download are about 57s and 52s respectively. There is no difference between individual and simultaneous boot up.

Boot-up times are about 90s and 76s respectively, when there is a four megabytes image do download and the robots are booting up individually. These times increase to 103s and 89s when the robots boot simultaneously. This can be explained by the fact that the robots have to share the wireless card bandwidth.

When there is a eight megabytes image to download, boot up times are approximately 102s and 85s respectively. These times increase to 138s and 125s when the robots boot simultaneously. Again, this can be explained by the fact that the robots have to share the wireless card bandwidth.

Average Boot-up Time				
	486 DX2 33MHz		Pentium I 66MHz	
	Individual (Seconds)	Simultaneous (Seconds)	Individual (Seconds)	Simultaneous (Seconds)
0MB	57.31	57.41	52.77	52.74
4MB	90.04	103.12	76.28	89.03
8MB	101.99	137.80	85.21	125.15

Average Download Time				
	486 DX2 33MHz		Pentium I 66MHz	
	Individual (Seconds)	Simultaneous (Seconds)	Individual (Seconds)	Simultaneous (Seconds)
0MB	0.00	0.00	0.00	0.00
4MB	12.57	23.69	8.32	21.22
8MB	24.88	58.75	17.38	57.42

Average Download Speed				
	486 DX2 33MHz		Pentium I 66MHz	
	Individual (MB/s)	Simultaneous (MB/s)	Individual (MB/s)	Simultaneous (MB/s)
0MB	0.00	0.00	0.00	0.00
4MB	325.83	172.89	492.49	193.03
8MB	329.25	139.44	471.35	142.66

Test with robots booting up individually and no image to download																		
	Robobug			Anic			Apé			Bist			Ch'up			Toms'up		
	Total time (seconds)	Download (seconds)	Voltage (volts)	Total time (seconds)	Download (seconds)	Voltage (volts)	Total time (seconds)	Download (seconds)	Voltage (volts)	Total time (seconds)	Download (seconds)	Voltage (volts)	Total time (seconds)	Download (seconds)	Voltage (volts)	Total time (seconds)	Download (seconds)	Voltage (volts)
1	57.41	n-a	12.44	52.88	n-a	12.36	52.34	n-a	12.30	53.00	n-a	12.33	57.72	n-a	12.01	53.05	n-a	12.31
2	57.59	n-a	12.42	52.88	n-a	12.36	53.00	n-a	12.25	52.84	n-a	12.30	58.06	n-a	11.95	52.21	n-a	12.29
3	56.88	n-a	12.41	53.09	n-a	12.35	52.91	n-a	12.16	52.94	n-a	12.26	57.75	n-a	11.94	52.68	n-a	12.28
4	57.59	n-a	12.38	53.09	n-a	12.33	52.15	n-a	12.09	52.62	n-a	12.20	58.37	n-a	11.91	52.01	n-a	12.27
5	57.00	n-a	12.36	53.19	n-a	12.31	52.13	n-a	12.07	52.38	n-a	12.16	57.66	n-a	11.89	52.98	n-a	12.25
6	57.72	n-a	12.35	53.34	n-a	12.30	52.38	n-a	12.07	52.28	n-a	12.16	58.16	n-a	11.89	53.02	n-a	12.23
7	57.06	n-a	12.34	53.06	n-a	12.28	52.76	n-a	12.07	52.75	n-a	12.16	57.71	n-a	11.82	52.54	n-a	12.22
8	57.13	n-a	12.32	53.62	n-a	12.26	52.75	n-a	12.06	52.60	n-a	12.16	57.72	n-a	11.79	52.16	n-a	12.21
9	57.56	n-a	12.31	53.44	n-a	12.24	52.69	n-a	12.06	53.03	n-a	12.16	58.59	n-a	11.76	52.78	n-a	12.21
10	57.16	n-a	12.29	53.04	n-a	12.23	52.44	n-a	12.06	52.94	n-a	12.15	57.85	n-a	11.73	52.98	n-a	12.20
Average	**57.31**			**53.16**			**52.56**			**52.74**			**57.96**			**52.64**		
Variance	**0.088**			**0.057**			**0.095**			**0.069**			**0.104**			**0.154**		

Test with robots booting up simultaneously and no image to download																		
	Robobug			Anic			Apé			Bist			Ch'up			Toms'up		
	Total time (seconds)	Download (seconds)	Voltage (volts)	Total time (seconds)	Download (seconds)	Voltage (volts)	Total time (seconds)	Download (seconds)	Voltage (volts)	Total time (seconds)	Download (seconds)	Voltage (volts)	Total time (seconds)	Download (seconds)	Voltage (volts)	Total time (seconds)	Download (seconds)	Voltage (volts)
1	57.33	n-a	12.21	53.12	n-a	12.30	52.36	n-a	12.20	52.98	n-a	12.25	57.46	n-a	12.24	52.26	n-a	12.12
2	57.68	n-a	12.20	52.98	n-a	12.30	52.98	n-a	12.29	52.15	n-a	12.24	57.87	n-a	12.23	53.01	n-a	12.11
3	56.98	n-a	12.20	53.12	n-a	12.29	52.61	n-a	12.27	52.48	n-a	12.22	57.84	n-a	12.21	52.76	n-a	12.10
4	57.76	n-a	12.19	53.41	n-a	12.28	52.48	n-a	12.26	52.68	n-a	12.21	58.03	n-a	12.20	52.09	n-a	12.09
5	57.43	n-a	12.18	53.24	n-a	12.27	52.18	n-a	12.25	52.94	n-a	12.20	57.61	n-a	12.19	52.48	n-a	12.08
6	57.12	n-a	12.17	53.64	n-a	12.27	52.21	n-a	12.24	52.47	n-a	12.19	57.44	n-a	12.17	52.69	n-a	12.07
7	57.91	n-a	12.15	53.18	n-a	12.26	52.78	n-a	12.22	52.26	n-a	12.17	57.81	n-a	12.16	52.47	n-a	12.06
8	57.46	n-a	12.14	53.16	n-a	12.25	52.61	n-a	12.21	52.73	n-a	12.16	57.67	n-a	12.14	52.43	n-a	12.05
9	57.31	n-a	12.13	53.45	n-a	12.24	52.12	n-a	12.20	52.76	n-a	12.15	57.98	n-a	12.13	52.76	n-a	12.04
10	57.16	n-a	12.12	53.36	n-a	12.23	52.48	n-a	12.19	52.73	n-a	12.13	58.24	n-a	12.12	52.81	n-a	12.02
Average	**57.41**			**53.27**			**52.48**			**52.62**			**57.80**			**52.58**		
Variance	**0.088**			**0.039**			**0.076**			**0.075**			**0.065**			**0.078**		

Test with robots booting up individually and a four MegaBytes image to download

	Robobug			Anic			Apé			Bist			Ch'up			Toms'up		
	Total time (seconds)	Download (seconds)	Voletge (volts)	Total time (seconds)	Download (seconds)	Voletge (volts)	Total time (seconds)	Download (seconds)	Voletge (volts)	Total time (seconds)	Download (seconds)	Voletge (volts)	Total time (seconds)	Download (seconds)	Voletge (volts)	Total time (seconds)	Download (seconds)	Voletge (volts)
1	91.88	13.51	12.45	79.44	10.80	12.30	78.10	9.81	12.25	77.60	9.94	12.35	82.37	8.67	12.16	77.85	9.06	12.31
2	90.71	13.48	12.43	76.28	7.82	12.28	75.40	8.16	12.19	75.87	8.34	12.31	81.03	8.42	12.16	76.53	8.82	12.29
3	89.63	12.77	12.41	76.28	8.04	12.23	75.29	7.68	12.13	75.75	7.86	12.27	81.38	8.13	12.14	76.12	8.02	12.26
4	89.03	12.17	12.39	76.78	7.88	12.19	75.85	7.99	12.10	76.25	8.25	12.22	80.69	8.74	12.12	76.23	8.12	12.24
5	90.28	12.28	12.38	76.48	8.04	12.13	75.56	8.01	12.04	77.09	9.79	12.16	81.34	7.61	12.10	77.06	7.94	12.21
6	90.40	12.70	12.36	76.85	8.53	12.50	75.96	8.45	12.03	75.57	8.13	12.11	80.66	7.89	12.08	76.38	8.01	12.19
7	89.09	12.08	12.33	75.62	7.94	12.34	76.28	8.30	12.00	75.97	8.24	12.08	80.46	8.88	12.05	76.24	8.23	12.17
8	90.03	12.17	12.32	76.00	7.55	12.22	75.63	8.13	11.98	74.93	8.09	12.04	81.75	8.88	12.02	75.98	8.21	12.15
9	90.06	12.36	12.30	76.56	7.89	12.15	75.88	7.96	11.96	75.97	7.99	12.01	81.46	8.62	12.00	76.02	8.45	12.13
10	89.29	12.19	12.28	76.00	7.67	12.13	75.19	7.68	11.94	75.50	8.30	12.00	80.94	8.21	11.99	76.67	8.56	12.10
Average	**90.04**	**12.57**		**76.63**	**8.22**		**75.91**	**8.22**		**76.05**	**8.49**		**81.21**	**8.41**		**76.51**	**8.34**	
Variance	**0.742**	**0.289**		**1.117**	**0.892**		**0.700**	**0.372**		**0.605**	**0.545**		**0.332**	**0.188**		**0.329**	**0.140**	

Test with robots booting up simultaneously and a four MegaBytes image to download

	Robobug			Anic			Apé			Bist			Ch'up			Toms'up		
	Total time (seconds)	Download (seconds)	Voletge (volts)	Total time (seconds)	Download (seconds)	Voletge (volts)	Total time (seconds)	Download (seconds)	Voletge (volts)	Total time (seconds)	Download (seconds)	Voletge (volts)	Total time (seconds)	Download (seconds)	Voletge (volts)	Total time (seconds)	Download (seconds)	Voletge (volts)
1	102.15	21.72	12.28	90.98	22.16	12.36	89.91	21.01	12.44	89.68	21.16	12.36	95.65	21.32	12.38	89.56	22.08	12.31
2	105.41	26.56	12.28	89.25	21.06	13.34	88.97	20.73	12.44	88.95	20.89	12.34	94.87	20.78	12.36	88.25	20.54	12.29
3	101.28	22.67	12.27	88.06	20.98	12.32	87.97	21.16	12.42	89.12	21.36	12.31	93.68	21.64	12.32	89.03	21.17	12.27
4	102.38	22.09	12.25	89.12	21.62	12.30	88.40	20.62	12.40	89.21	21.45	12.29	93.98	20.16	12.29	88.98	21.72	12.24
5	103.87	23.20	12.23	89.75	21.85	12.28	87.69	20.35	12.36	88.32	20.98	12.26	93.67	20.24	12.27	88.53	20.86	12.21
6	102.44	23.59	12.22	90.32	22.85	12.24	90.68	22.49	12.34	88.65	21.56	12.24	94.12	22.32	12.25	88.57	21.64	12.19
7	102.88	25.27	12.15	88.12	20.06	12.22	89.94	22.13	12.31	88.06	21.49	12.21	93.32	21.12	12.22	89.25	22.29	12.16
8	103.37	23.44	12.12	89.98	20.16	12.19	89.28	22.82	12.28	88.65	21.96	12.19	94.56	22.89	12.20	89.65	21.73	12.13
9	103.62	24.60	12.09	87.06	18.98	12.17	89.06	19.82	12.25	89.56	21.78	12.17	93.45	21.67	12.18	90.12	21.59	12.10
10	103.75	23.77	12.06	89.56	20.69	12.15	86.06	18.53	12.21	90.45	21.78	12.15	94.16	22.43	12.14	88.42	20.67	12.08
Average	**103.12**	**23.69**		**89.22**	**21.04**		**88.80**	**20.97**		**89.07**	**21.44**		**94.15**	**21.46**		**89.04**	**21.43**	
Variance	**1.323**	**2.151**		**1.390**	**1.292**		**1.776**	**1.662**		**0.499**	**0.124**		**0.511**	**0.842**		**0.372**	**0.352**	

Test with robots booting up individually and an eight MegaBytes image to download

	Robobug			Anic			Apé			Bist			Ch'up			Toms'up		
	Total time (seconds)	Download (seconds)	Voletge (volts)	Total time (seconds)	Download (seconds)	Voletge (volts)	Total time (seconds)	Download (seconds)	Voletge (volts)	Total time (seconds)	Download (seconds)	Voletge (volts)	Total time (seconds)	Download (seconds)	Voletge (volts)	Total time (seconds)	Download (seconds)	Voletge (volts)
1	102.47	25.30	12.26	89.63	20.91	12.25	87.22	19.62	12.45	90.44	22.71	12.55	93.69	20.78	12.48	89.91	20.12	12.41
2	101.72	24.21	12.26	86.19	17.31	12.23	82.41	15.65	12.43	85.13	17.22	12.53	91.02	17.46	12.47	86.56	17.59	12.39
3	101.44	24.00	12.25	84.94	17.09	12.22	84.38	16.95	12.39	86.94	19.46	12.51	89.76	16.81	12.45	85.01	17.08	12.37
4	102.34	25.35	12.23	84.43	16.42	12.20	84.56	16.47	12.35	84.91	17.44	12.48	90.45	16.45	12.43	84.16	16.47	12.34
5	101.97	24.60	12.21	86.63	18.48	12.17	86.34	18.12	12.31	84.06	16.84	12.43	91.72	17.01	12.42	85.59	17.49	12.31
6	101.91	25.33	12.19	84.16	16.38	12.14	84.31	18.12	12.26	83.91	16.00	12.41	90.78	16.92	12.38	85.76	17.58	12.28
7	102.44	25.25	12.16	85.97	17.30	12.11	84.25	16.50	12.22	86.16	17.94	12.36	89.86	16.72	12.33	84.02	16.89	12.24
8	102.37	25.58	12.14	83.50	15.99	12.08	85.81	18.09	12.17	84.31	16.25	12.31	89.12	16.43	12.29	85.19	17.65	12.21
9	102.41	25.04	12.12	83.22	14.20	12.05	84.62	16.75	12.12	82.81	15.94	12.25	90.16	17.21	12.26	84.59	17.58	12.18
10	100.84	24.15	12.09	83.01	14.71	12.02	84.22	17.02	12.08	84.90	17.65	12.21	90.12	17.18	12.22	84.12	17.21	12.15
Average	**101.99**	**24.88**		**85.17**	**16.88**		**84.81**	**17.33**		**85.36**	**17.75**		**90.67**	**17.30**		**85.49**	**17.57**	
Variance	**0.287**	**0.343**		**4.065**	**3.595**		**1.789**	**1.322**		**4.534**	**4.152**		**1.651**	**1.605**		**3.078**	**0.951**	

Test with robots booting up simultaneously and an eight MegaBytes image to download

	Robobug			Anic			Apé			Bist			Ch'up			Toms'up		
	Total time (seconds)	Download (seconds)	Voletge (volts)	Total time (seconds)	Download (seconds)	Voletge (volts)	Total time (seconds)	Download (seconds)	Voletge (volts)	Total time (seconds)	Download (seconds)	Voletge (volts)	Total time (seconds)	Download (seconds)	Voletge (volts)	Total time (seconds)	Download (seconds)	Voletge (volts)
1	151.44	73.53	12.55	130.12	61.58	12.36	129.34	61.06	12.21	130.21	61.49	12.20	136.21	61.26	12.28	129.87	61.28	12.16
2	133.38	52.98	12.52	121.06	54.87	12.35	117.65	49.81	12.21	122.25	53.87	12.18	126.12	53.97	12.27	120.64	51.64	12.15
3	132.60	53.59	12.49	119.87	51.21	12.33	119.03	50.88	12.18	120.03	51.87	12.16	125.18	52.06	12.25	122.61	54.59	12.14
4	131.09	52.65	12.43	124.36	57.68	12.31	127.84	59.68	12.15	125.64	58.49	12.14	131.19	58.68	12.23	124.56	58.03	12.12
5	131.25	51.71	12.40	124.68	58.16	13.29	124.87	58.32	12.11	123.45	54.26	12.13	128.78	54.63	12.21	124.87	57.87	12.10
6	147.13	67.62	12.35	126.89	59.16	12.27	119.09	50.56	12.08	126.67	60.18	12.10	132.14	60.59	12.19	129.01	61.89	12.08
7	130.41	51.25	12.31	128.95	61.87	12.16	128.31	59.95	12.03	128.75	58.27	12.08	134.36	59.68	12.17	126.42	60.91	12.06
8	142.43	63.05	12.26	122.36	53.98	12.24	125.37	58.43	12.00	121.35	52.86	12.06	126.87	53.12	12.15	125.24	59.12	12.04
9	144.59	65.18	12.15	125.13	58.27	12.12	129.47	62.26	11.95	126.45	60.25	12.04	132.57	61.24	12.13	122.67	54.16	12.01
10	133.69	55.93	12.12	128.76	60.58	12.21	124.06	56.64	11.92	129.26	60.87	12.02	135.64	61.46	12.10	128.89	60.12	11.99
Average	**137.80**	**58.75**		**125.22**	**57.74**		**124.50**	**56.76**		**125.41**	**57.24**		**130.91**	**57.67**		**125.48**	**57.96**	
Variance	**60.739**	**63.095**		**12.043**	**11.929**		**20.111**	**21.555**		**12.333**	**13.314**		**15.928**	**14.299**		**9.453**	**11.881**	

Appendix E -

Source Code of PIC Boot-Loading

```
;*******************************************************************************
;*******************************************************************************
;          Boot - and Interrupt loader for Microchip PIC16F87X
;*******************************************************************************
;*******************************************************************************
; Name of file:        Re_Embodiment.asm
; Date:                12.12.2002
; Author:              Jerome Corre
; University:          Cardiff  University
; Email:               correj1@cf.ac.uk
; based on:            Michael Cummins and Ekachai Asawabunsap (loader.asm)
; Email:               cumminsm@cf.ac.uk
;*******************************************************************************
;*******************************************************************************
; Use in connection with PICdownloader 1.08 for windows or linux.
; For interrupt loading User Program Interrupt Template is required (template.asm)
;*******************************************************************************
;*******************************************************************************

          errorlevel  -302, -306                ; no message 302 and 306
          list        b=2                       ; tabulator size = 2

;================== User setting section ========================================

          list p=16f877                         ; type of micro-controller
#include   <p16f877.inc>
```

```
#define    ProgHI       0x1FFF
#define    FOSC         D'4000000'                  ; quartz frequency [ Hz], max. 20 MHz
#define    BAUD         D'9600'                     ; baud rate [ bit/sec] (19200/9600 for 20/4MHz)
#define    TIMEOUT      D'1'                        ; time [ 0.1s], max. 25 sec (D'254')

;================== Configuration ==========================================

           __IDLOCS     H'2100'                     ; version ID of bootloader
           __CONFIG _CP_OFF & _WDT_OFF & _BODEN_OFF & _PWRTE_ON & _HS_OSC & _WRT_ENABLE_ON & _LVP_ON & _DEBUG_OFF & _CPD_OFF

;========================= Constants ======================================

#define    DIVIDER      (FOSC/(D'16' * BAUD))-1     ; required for baud rate
#define    HIGH_SPEED   1                           ; high speed on
#define    T1PS         8                           ; Timer configuration
#define    T1SU         0x31
TIMER      EQU          (D'65538'-(FOSC/(D'10'*4*T1PS)))    ; reload value for TIMER1 (0.1s int)

#define    LoaderSize   0xFF                        ; size of bootloader
#define    LoaderMain   UserStart+5                 ; main address of bootloader
#define    LoaderTop    ProgHI                      ; top address of bootloader
#define    LoaderStart  (LoaderTop)-LoaderSize+1    ; start address of bootloader

#define    NumRetries   1                           ; number of writing retries
```

```
#define    WRITE       0xE3                    ; write
#define    WR_OK       0xE4                    ; Write ok
#define    WR_BAD      0xE5                    ; write bad
#define    DATA_OK     0xE7                    ; received data ok
#define    DATA_BAD    0xE8                    ; received data bad
#define    IDACK       0xEB                    ; PIC -> PC ID-Acknowledgement
#define    DONE        0xED                    ; job done
#define    Twait       D'150'                  ; Time for Interrupt Timeout
#define    ID_ALL      0xEA                    ; PC -> PIC Identifier for all robots
#define    IDENT       0xEB                    ; PC -> PIC Identifier for robot 'XX'

;================= Variables ================================================

buff       EQU         0x20
amount     EQU         0x71
chk1       EQU         0x72
chk2       EQU         0x73
retry      EQU         0x74
address    EQU         0x75
tmpaddr    EQU         0x77
temp       EQU         0x79
time       EQU         0x7A
count      EQU         0x7B

twait1     EQU         0x80                    ; Time variable for Interrupt Timeout
twait2     EQU         0x81                    ; Time variable for Interrupt Timeout
```

```
;------------------ reset vector ----------------------------------------------------------------------------
          ORG         0x0000                    ; start of the boot code
          Pagesel     Main
          Goto        Main                      ; jumps to main boot code in upper memory

;------------------ interrupt vector ------------------------------------------------------------------------
          ORG         0x0004                    ; user program interrupt vector
          Pagesel     interrupt
          goto        interrupt                 ; goto Interrupt Service Routine

;------------------ user reset code -------------------------------------------------------------------------
          ORG         LoaderStart
TrapError
          Pagesel     TrapError
          goto        TrapError                 ; trap for unintended running into Bootloader

UserStart
; this instruction never gets overwritten
          clrf        PCLATH                    ; clear PCLATH and change to bank 0

; the following 2 instructions get overwritten by user program
          pagesel     UserStart                 ; set PCLATH to program page of UserStart
          goto        UserStart                 ; loop for first start without a user program
; If first 2 relocated instructions of user code
;don't contain a branch to Main, execution
```

```
;remains in endless loop to avoid unintended
;operation
;------------------- start of bootloading code ------------------------------------------------------------------------
          ORG         LoaderMain                    ; LoaderMain has address of UserStart+5
Main
;------------------- setup of USART -----------------------------------------------------------------------------------

;set up USART for Asynchronous communication

          movlw       0x90                          ; SPEN = 1, CREN = 1
          movwf       RCSTA                         ;move contents of w reg to receive status and control register
          bsf         STATUS,RP0                    ; set to bank1
                                                    ; USART SYNC=0; SPEN=1; CREN=1; SREN=0;
          bsf         TXSTA,BRGH                    ; TX9=0; RX9=0; TXEN=1; (high baud rate enabled)
          bsf         TXSTA,TXEN                    ; transmission enabled
          movlw       DIVIDER                       ; baud rate generator
          movwf       SPBRG

;------------------------- TIMER ---------------------------------------------------------------------------------------
timer
          bcf         STATUS,RP0
          movlw       TIMEOUT+1                     ; set timeout
          movwf       time
          movlw       T1SU
          movwf       T1CON                         ; TIMER1 on, internal clock, prescale T1PS
          bsf         PIR1,TMR1IF
```

```
          call      getbyte                 ; wait for IDENT or ID_ALL
          xorlw     IDENT                   ; call getbyte subroutine
          btfsc     STATUS,Z                ; skip when IDENT was not received
          goto      cl_time                 ; go to clear time and download
          movf      RCREG,w                 ; RCREG
          xorlw     ID_ALL
          btfss     STATUS,Z                ; skip when ID_ALL was received
          goto      user_restore
cl_time   clrf      time                    ; no more wait for IDENT
          goto      inst_ident              ; bootloader identified, send of IDACK (ID Acknowledgement)


;---------------------- RECEIVE ----------------------------------------------------------------------------------
receive                                     ; programming
          call      getbyte                 ; get byte from USART
          movwf     temp
          xorlw     WRITE
          btfsc     STATUS,Z
          goto      inst_write              ; write instruction

          movf      temp,w
          xorlw     DONE
          btfss     STATUS,Z                ; done instruction ?
          goto      receive

;-----------------------------------------------------------------------------------------------------------------
```

```
inst_done                                   ; very end of programming
;-------------------------------------------------------------------------------------------------------------------
          movlw     WR_OK
call      putbyte                           ; send of byte
          movlw     TIMEOUT+1
          movwf     time
call      getbyte                           ; has built in timeout - waits until done
;-------------------------------------------------------------------------------------------------------------------
user_restore
          clrf      T1CON                   ; shuts off TIMER1
          clrf      RCSTA
          bsf       STATUS,RP0
          clrf      TXSTA                   ; restores USART to reset condition
          bcf       STATUS,RP0
          clrf      PIR1
          goto      UserStart               ; run user program



;-------------------------------------------------------------------------------------------------------------------
inst_ident
          movlw     IDACK                   ; send IDACK
          goto      send_byte
;-------------------------------------------------------------------------------------------------------------------
inst_write
          call      getbyte
```

```
            movwf       address+1                   ; high byte of address
            call        getbyte
            movwf       address                     ; low byte of address
            call        getbyte
            movwf       amount                      ; number of bytes -> amount -> count
            movwf       count
            call        getbyte                     ; checksum -> chk2
            movwf       chk2
            clrf        chk1                        ; chk1 = 0
            movlw       buff
            movwf       FSR                         ; FSR pointer = buff
receive_data
            call        getbyte                     ; receive next byte -> buff[ FSR]
            movwf       INDF
            addwf       chk1,f                      ; chk1 := chk1 + buff[ FSR]
            incf        FSR,f                       ; FSR++
            decfsz      count,f
            goto        receive_data                ; repeat until (--count==0)
checksum
            movf        chk1,w
            xorwf       chk2,w                      ; if (chk1 != chk2)
            movlw       DATA_BAD
            btfss       STATUS,Z
            goto        send_byte                   ; checksum WRONG
checksum_ok
            movlw       DATA_OK                     ; checksum OK
```

```
        call    putbyte
write_byte
        call    write_eeprom            ; write to eeprom
        iorlw   0
        movlw   WR_OK                   ; writing OK
        btfsc   STATUS,Z
        movlw   WR_BAD                  ; writing WRONG

;-------------------------------------------------------------------------------------------------------------
send_byte
        call    putbyte                 ; send of byte
        goto    receive                 ; go to receive from UART
;-------------------------------------------------------------------------------------------------------------

;----------------------- putbyte subroutine ----------------------------------------------------------------
putbyte
        clrwdt                          ; clear watchdog timer
        btfss   PIR1,TXIF               ; check if Transmit Interrupt Flag bit of PIR1
                                        ; register is set, if yes, transmit buffer is empty,
                                        ; skip next line
        goto    putbyte                 ; if transmit buffer full goto putbyte
        movwf   TXREG                   ; move byte to transmit register of USART
        return

;----------------------- getbyte subroutine ----------------------------------------------------------------
```

```
getbyte
          movf      time,w                  ; move time value to w register
          btfsc     STATUS,Z                ; check for time==0
          goto      getbyte3                ; jump to getbyte 3
          btfss     PIR1,TMR1IF             ; check for TIMER1 overflow
          goto      getbyte3                ; no overflow
          bcf       T1CON,TMR1ON            ; timeout 0.1 sec
          decfsz    time,f                  ; time--
          goto      getbyte2
          retlw     0                       ; if time==0 then return
getbyte2
          bcf       PIR1,TMR1IF
          movlw     high TIMER
          movwf     TMR1H                   ; pre-set TIMER1 for 0.1s timeout
          bsf       T1CON,TMR1ON
getbyte3
          btfss     PIR1,RCIF               ; while(!RCIF)
          goto      getbyte
          movf      RCREG,w                 ; RCREG
          return

;------------------------ write eeprom subroutine ------------------------------------------------------------

write_eeprom
          movf      address,w
          movwf     tmpaddr                 ; tmpaddr = address
```

```
        movf    address+1,w
        movwf   tmpaddr+1
        clrf    count                   ; count=0
write_loop
        movlw   NumRetries+1            ; retry = NumRetries+1
        movwf   retry
w_e_l_1
        movf    amount,w
        subwf   count,w                 ; while (count<amount)
        btfsc   STATUS,C
        retlw   1                       ; otherwise return 1 (OK)
        movf    count,w
        addlw   buff                    ; set buffer pointer
        movwf   FSR
w_e_l_2
        movlw   0x21                    ; if (0x2100 <= tmpaddr <= 0x21FF)
        subwf   tmpaddr+1,w
        bsf     STATUS,RP1
        bsf     STATUS,RP0              ; (bank3)
        btfsc   STATUS,Z
        goto    data_eeprom             ; goto data_eeprom
program_eeprom
        bsf     EECON1,EEPGD            ; EEPGD = 1 -> program memory
        clrf    STATUS
        movlw   high (LoaderStart)      ; if (tmpaddr >= LoaderStart)
        subwf   tmpaddr+1,w
```

```
        movlw   low (LoaderStart)       ; mask Bootloader, [ ICD-Debugger],
        btfsc   STATUS,Z                ; __IDLOCS & __CONFIG
        subwf   tmpaddr,w
        btfsc   STATUS,C
        goto    next_adr                ; next address
        goto    w_e_l_3
data_eeprom
        bcf     EECON1,EEPGD            ; EEPGD = 0 -> data memory
        clrf    STATUS
w_e_l_3
        movf    tmpaddr,w
        bsf     STATUS,RP1
        movwf   EEADR                   ; EEADR = low tmpaddr
        bcf     STATUS,RP1
        movf    tmpaddr+1,w             ; if (tmpaddr < 0x0004)
        btfss   STATUS,Z
        goto    w_e_l_4
        movlw   4
        subwf   tmpaddr,w
        btfsc   STATUS,C
        goto    w_e_l_4
        bsf     STATUS,RP1              ; (bank3)
        bsf     STATUS,RP0
        btfss   EECON1,EEPGD            ; skip if (EEPGD)
        goto    w_e_l_31
        bcf     STATUS,RP0              ; (bank2)
```

```
        movlw   low UserStart+1         ; EEADRL + low UserStart+1
        addwf   EEADR,f                 ; (relocated first 4 user instructions)
w_e_l_31
        clrf    STATUS                  ; (bank0)
        movlw   high UserStart          ; EEADRH = high UserStart
        goto    w_e_l_5
w_e_l_4
        movf    tmpaddr+1,w             ; EEADRH = high tmpaddr
w_e_l_5
        bsf     STATUS,RP1
        movwf   EEADRH                  ; set EEADRH
        movf    INDF,w
        movwf   EEDATH                  ; EEDATH = buff[ count]
        incf    FSR,f
        movf    INDF,w
        movwf   EEDATA                  ; EEDATA = buff[ count+1]
        bsf     STATUS,RP0
        bsf     EECON1,WREN             ; WREN=1
        movlw   0x55                    ; EECON2 = 0x55
        movwf   EECON2
        movlw   0xAA                    ; EECON2 = 0xAA
        movwf   EECON2
        bsf     EECON1,WR               ; WR=1
        nop                             ; instructions are ignored
        nop                             ; micro-controller waits for a complete write
        clrf    STATUS
```

```
wait_write
          clrwdt
          btfss       PIR2,EEIF                  ; necessary for a write to data eeprom
          goto        wait_write
          bcf         PIR2,EEIF
          bsf         STATUS,RP0                 ; (bank3)
          bsf         STATUS,RP1
          bcf         EECON1,WREN                ; WREN=0
          bsf         EECON1,RD                  ; RD=1
          nop
          nop
          bcf         STATUS,RP0                 ; (bank2)
          decf        FSR,f
          movf        INDF,w                     ; if ((EEDATH != buff[ count] ) || (EEDATA !=
          ; buff[ count+1] ))
          xorwf       EEDATH,w
          btfss       STATUS,Z
          goto        w_e_l_6                    ; repeat write
          incf        FSR,f
          movf        INDF,w
          xorwf       EEDATA,w
          btfsc       STATUS,Z
          goto        next_adr                   ; verification OK, next address
w_e_l_6
          clrf        STATUS                     ; (bank0)
          decfsz      retry,f
```

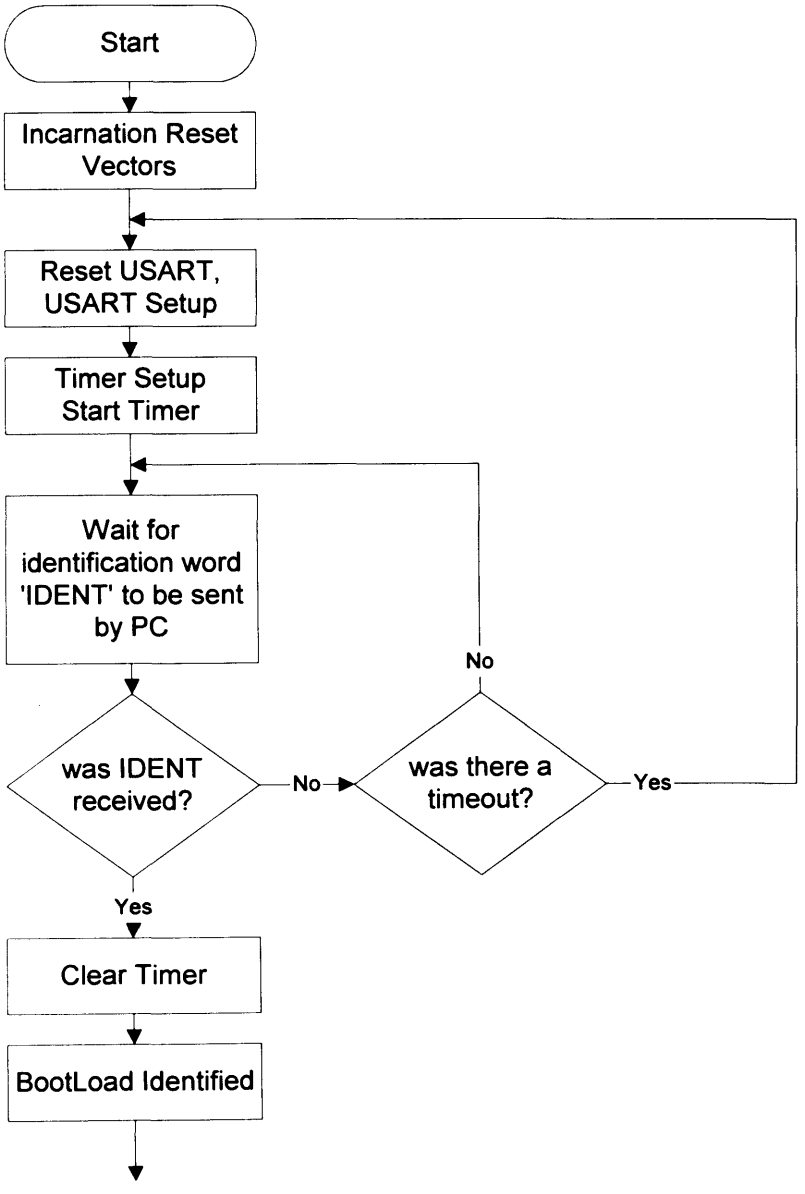
```
          goto      w_e_l_1                    ; if (--retry != 0) repeat write
          retlw     0                          ; else return 0 (BAD)
next_adr
          bcf       STATUS,RP1
          movlw     2                          ; count := count + 2
          addwf     count,f
          incf      tmpaddr,f                  ; tmpaddr := tmpaddr + 1
          btfsc     STATUS,Z
          incf      tmpaddr+1,f
          goto      write_loop


;--------------------- INTERRUPT SERVICE ROUTINE --------------------------------------------------------

interrupt                                      ; Interrupt Service Routine
          bcf       STATUS,RP0                 ; bank0
          bcf       STATUS,RP1

          btfsc     INTCON,INTF                ; external interrupt ?
          goto      UserInt                    ; if yes, goto user interrupt service routine

          movf      RCREG,w                    ; writes receive buffer to working register
          xorlw     IDENT                      ; if IDENT was received, xorlw=0 --> skip
          btfss     STATUS,Z                   ; else xorlw=1
          goto      All_Rob                    ; goto All_Rob
          goto      Download
```

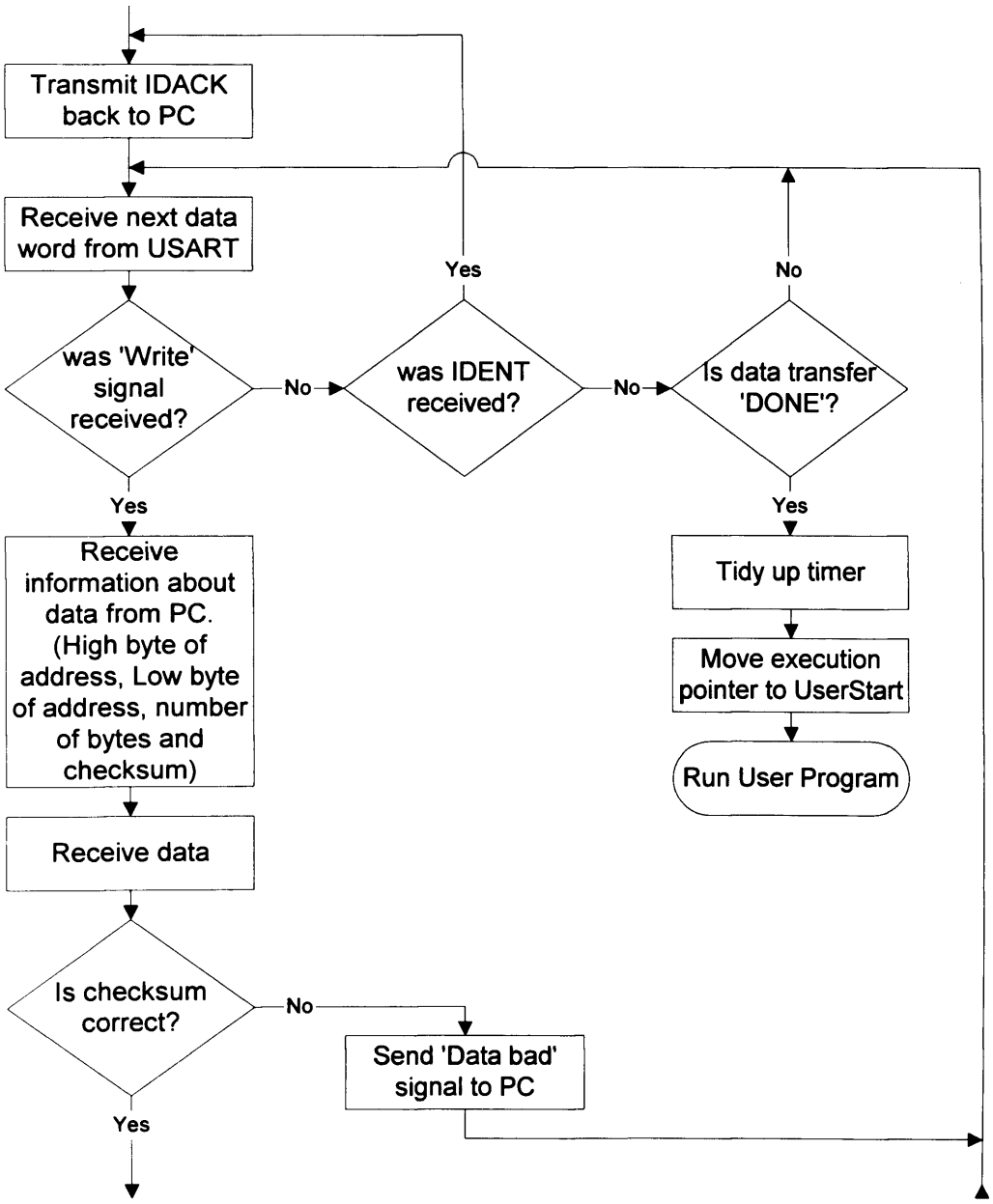
```
All_Rob    movf       RCREG,w                  ; writes receive buffer to working register
           xorlw      ID_ALL                   ; if ID_ALL was received, xorlw=0 --> skip
           btfss      STATUS,Z                 ; else xorlw=1
           goto       SerInt                   ; goto Serial Interrupt Service Routine

Download                                       ; if IDENT was received
           call       iwait                    ; required interrupt timeout
           pagesel    Main
           goto       Main                     ; goto main and download new user code

UserInt                                        ; if an external interrupt occurred (user interrupt)
           Pagesel    0x1E00                   ; go to user Interrupt Service Routine
           goto       0x1E00

SerInt                                         ; if any serial interrupt occurred - except IDENT
           bcf        RCSTA,CREN
           bsf        RCSTA,CREN               ; clear the overrun error bit OERR
           retfie                              ; return from interrupt to user program
;-------------------------------------------------------------------------------------------------------------
iwait      movlw      Twait                    ; Interrupt Timeout
           movwf      twait1
iwait11    decf       twait1,1
           call       iwait2
           btfss      STATUS,Z
           goto       iwait11
```

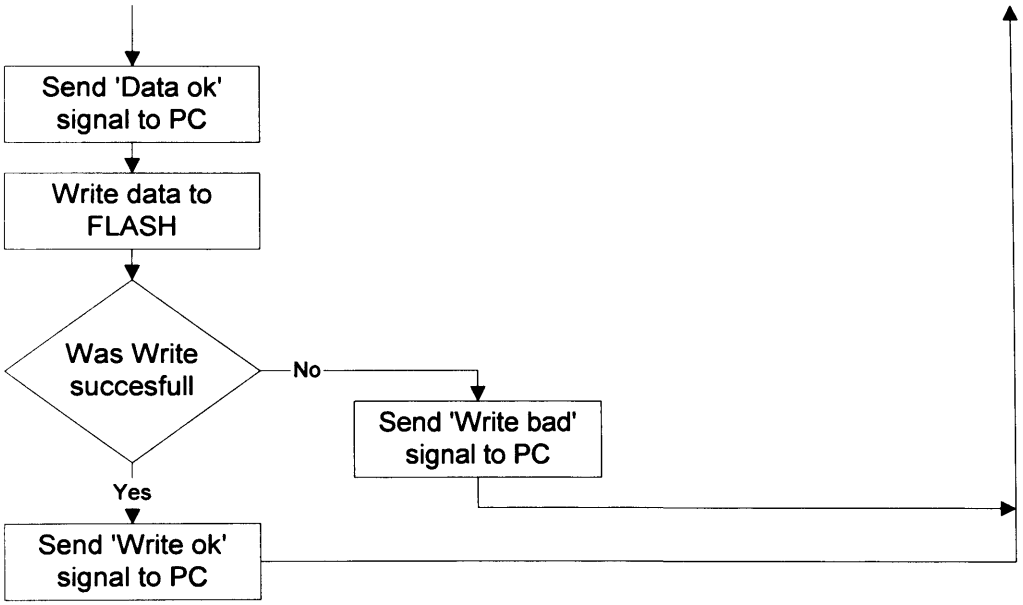
```
    return

iwait2  movlw   Twait
        movwf   twait2
iwait21 decfsz  twait2,1
        goto    iwait21

        return
;-----------------------------------------------------------------

        END
```

Appendix F -

PIC Incarnation Code Flowchart



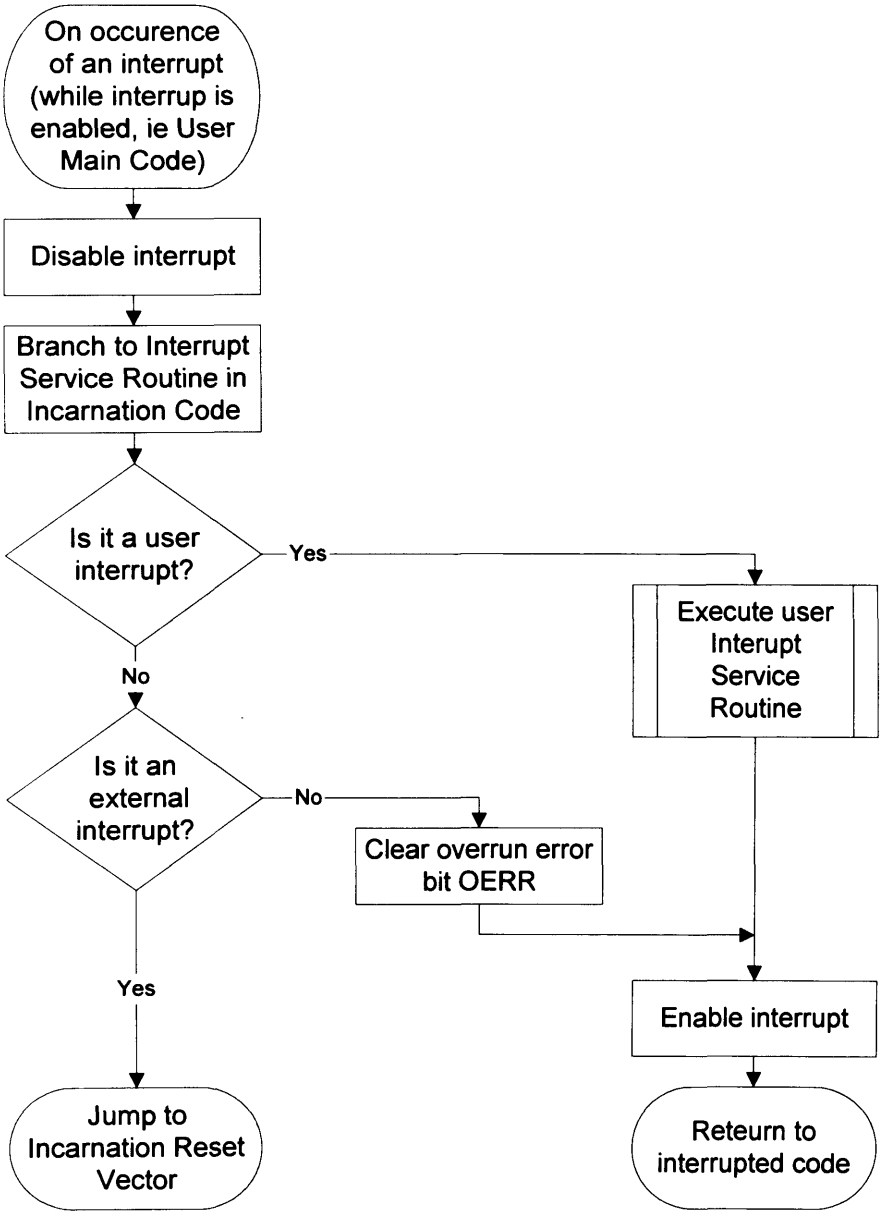

Transmit IDACK back to PC
Receive next data word from USART
was 'Write' signal received?
No
was IDENT received?
Yes
No
Is data transfer 'DONE'?
No
Yes
Yes
Receive information about data from PC. (High byte of address, Low byte of address, number of bytes and checksum)
Tidy up timer
Move execution pointer to UserStart
Run User Program
Receive data
Is checksum correct?
No
Send 'Data bad' signal to PC
Yes


Send 'Data ok' signal to PC
Write data to FLASH
Was Write succesfull
No
Send 'Write bad' signal to PC
Yes
Send 'Write ok' signal to PC

Appendix G -

PIC Interrupts Service Routine

Flowchart



Appendix H - ZoomCam Specifications

The ZoomCam PPC is an affordable, full-colour, parallel port camera (PPC). It is based on the VV0670 Vision CPiA chip **[Vision, 1998]**, which interface with the VV6404 digital CMOS sensor **[ST.Microelectronics, 1998]**. The Specifications of the CMOS sensor are detailed below. The Camera is fully supported under Windows™ by the manufacturer, under linux a third party driver is available[8].

VV6404 Coulour CMOS Sensor	
Image Format	352 x 288 pixels (CIF)
Pixel Size	12.0 x 11.0µm
Image Array Size	4.2mm x 3.2mm
Array Format	CIF
Exposure Control	25000:1
Sensor Signal/Noise Ratio	42dB
Supply Voltage	5.0v DC +/-5%
Package Type	48LCC
Operating Temp. Range	0oC - 40oC
Serial Interface Frequency Range	0-100kHz

[8] CPiA webcam driver for Linux (http://webcam.sourceforge.net/)

Appendix I - A Typical Calibration Sequence

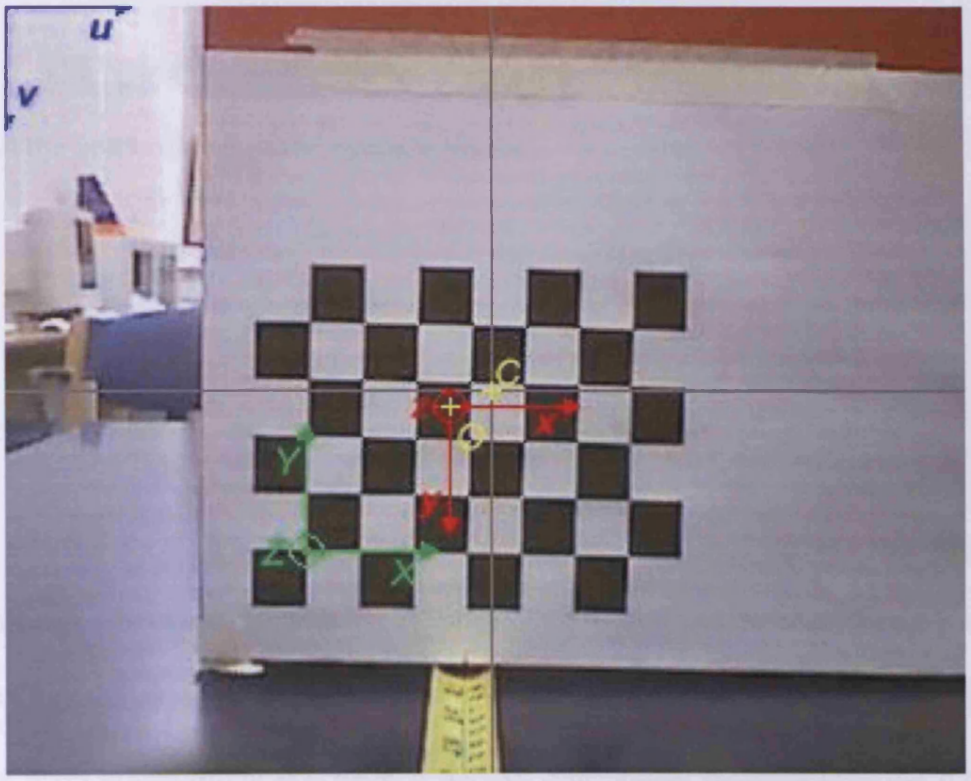
The following illustrates a typical calibration sequence for a robot's camera. Each images of the calibration chessboard taken during the sequence, are presented in turn, at he end of this section, in table, at the top of each columns. The second row illustrates the results of the corner approximation algorithm for each image. In the third row, the results of the refined sub pixel corner detection are presented. Finally in the fourth row displays, the original image undistorted, as computed from the calibration data. A total of seventeen images were taken in this particular sequence. Note that in the first image, the location and orientation of the chessboard, with respect to the robot reference frame, are known.

The intrinsic parameters derived from this calibration sequence are as follow:

$$P_{\text{int}} = \begin{bmatrix} 443.5 & 0 & 160.7 \\ 0 & 485.9 & 150.0 \\ 0 & 0 & 1 \end{bmatrix}$$

This places the principal point o slightly of from the centre of the image c, as can be expected from a cheaply manufactured camera. Both points o and c are illustrated in yellow on a copy of the first image of the calibration sequence, below.

$$o = \begin{bmatrix} 160.7 \\ 150.0 \end{bmatrix} \quad c = \begin{bmatrix} 176 \\ 144 \end{bmatrix}$$



The world reference frame (W,X,Y,Z), as defined by the chessboard for this frame, is represented in green. The camera reference frame (O,x,y,z) located at the focal point O is represented in red. The z axis of the camera reference frame is collinear with the focal axis and pierces the image at the principal point o. The image coordinate frame (i,u,v) is represented in blue. Note that the camera reference frame and the image coordinate frame have the same orientation.

Given that the pixel size of the camera is specified as:

$$S_x = 12\mu m \quad S_y = 11\mu m$$

One can thus derive the focal length as calculated along the horizontal and the vertical axes of the image's pixels:

$$f_x = \frac{f_1}{s_x} \text{ Thus } f_1 = 5.32mm$$

$$f_y = \frac{f_2}{s_y} \text{ Thus } f_2 = 5.34mm$$

The focal length is obviously a unique physical value, the slight difference between the two focal length calculation can be explained by the fact that the skew α_c was assumed to be 0, as part of the calibration. In reality the pixel edges might not be at a perfect right angle. Nevertheless, the focal length f will be assumed to be equal to the average of the two calculated values f_1 and f_2, thus giving:

$$f = 5.33mm$$

The extrinsic parameters derived from the calibration are:

$$P_{ext} = \begin{bmatrix} 0.999 & 0.020 & -0.009 & -53.15 \\ 0.021 & -0.998 & 0.051 & 50.32 \\ -0.008 & -0.052 & -0.998 & 456.17 \end{bmatrix}$$

The radial and tangential distortion parameters were estimated to be:

$$k_1 = -0.463 \quad k_2 = 0.656 \quad k_3 = 0.000$$

And:

$$p_1 = -0.011 \quad p_2 = 0.006$$

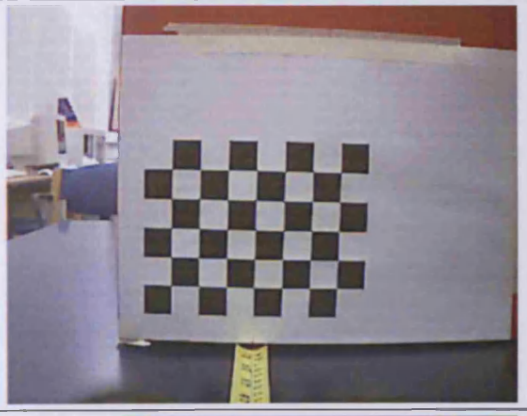
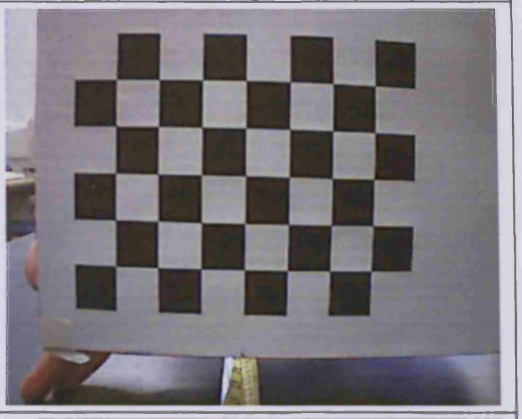
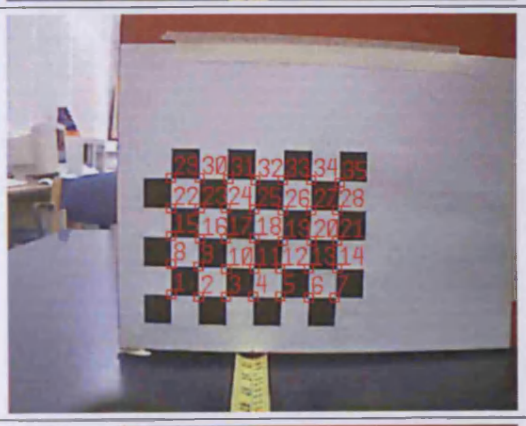
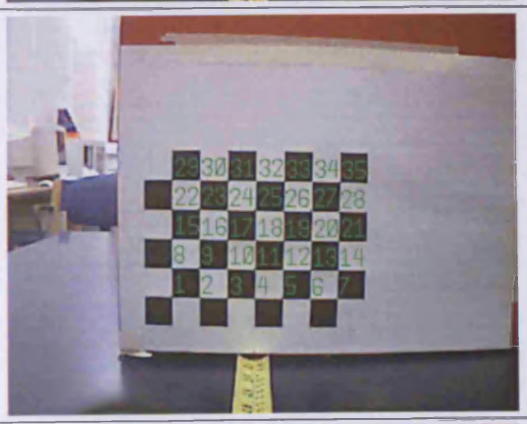
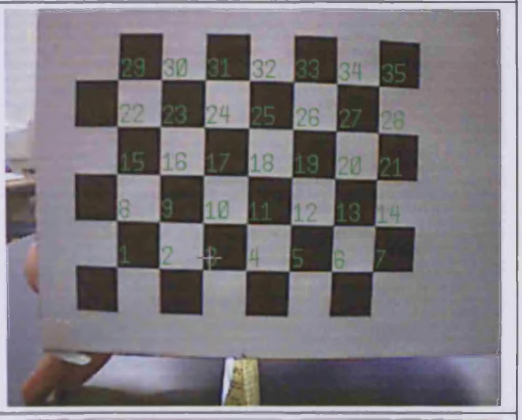
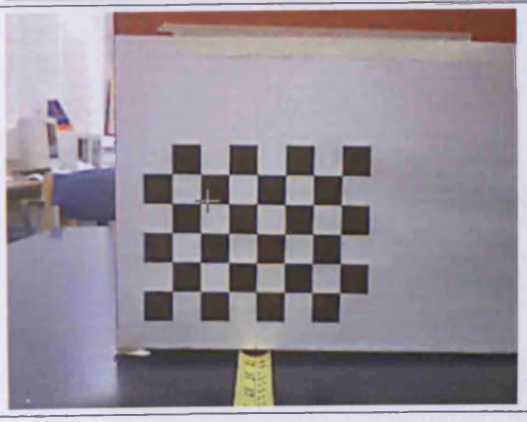
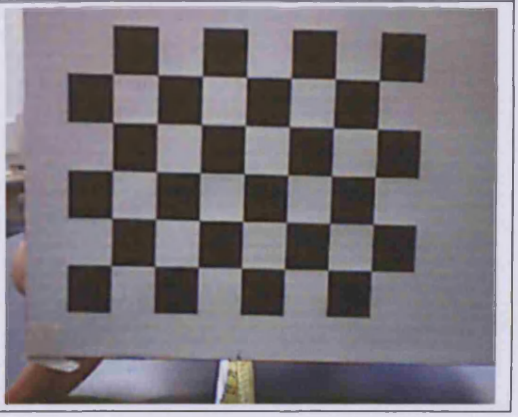
Image 1	Image 2
	
	
	
	

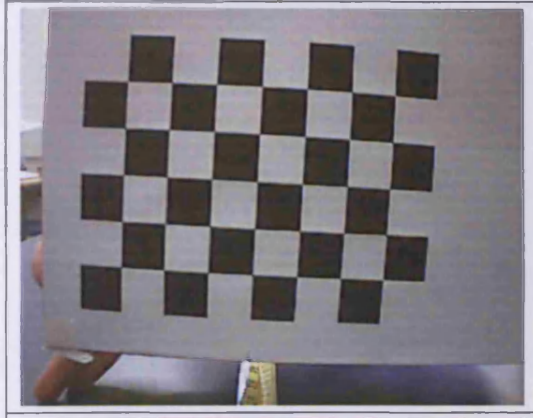
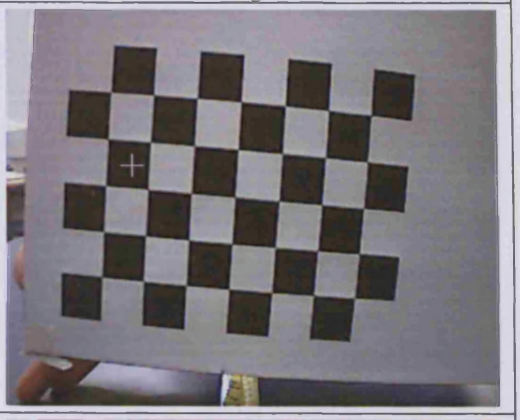
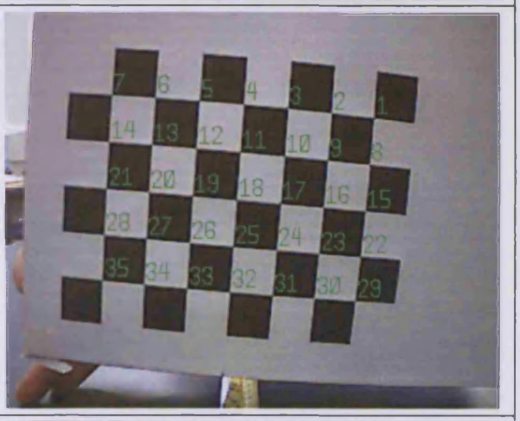
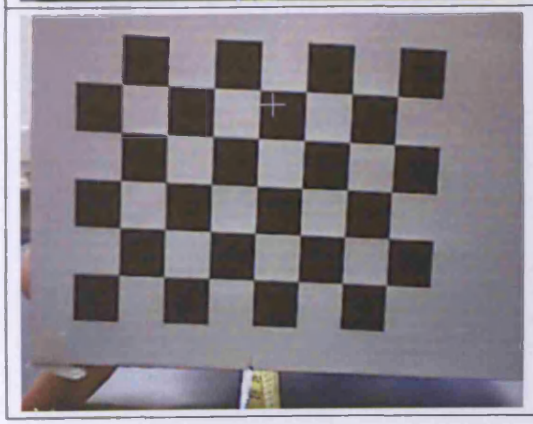
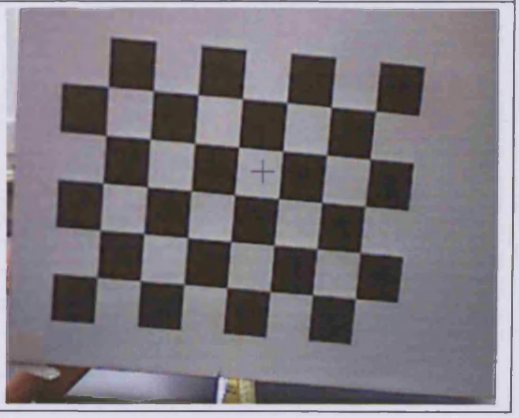
Image 3	Image 4
	
	
	
	

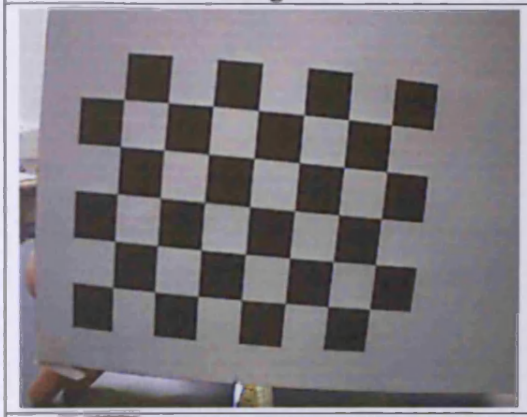
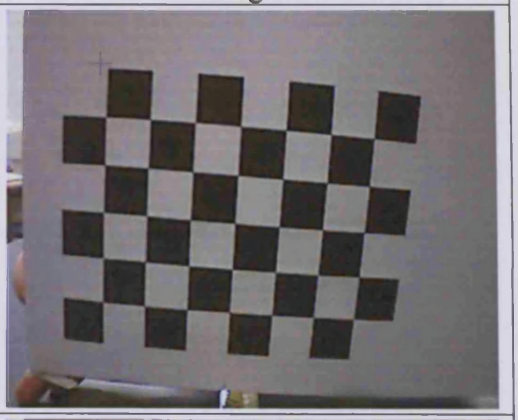
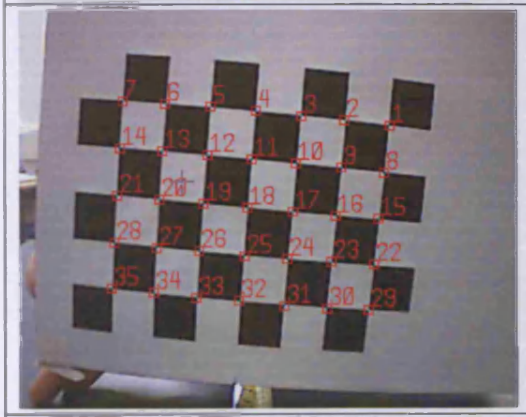
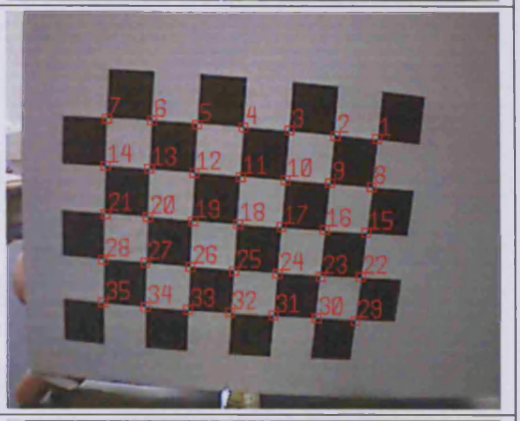
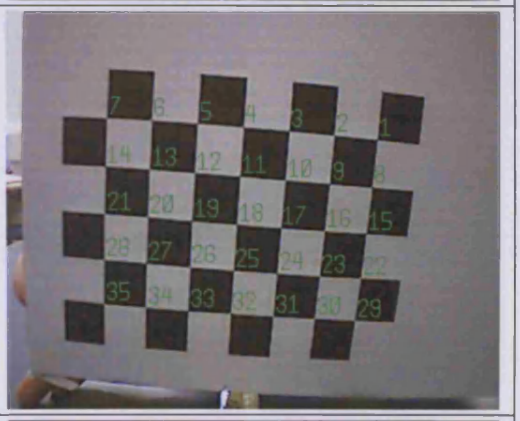
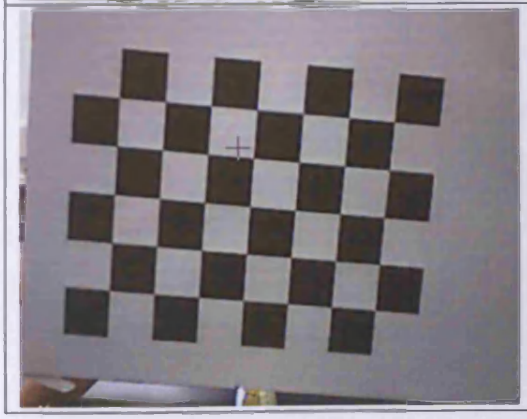
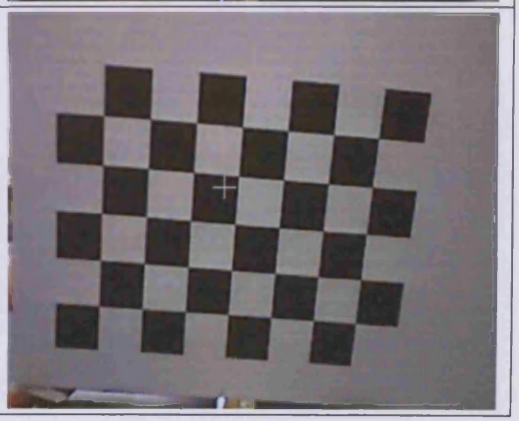
Image 5	Image 6
	
	
	
	

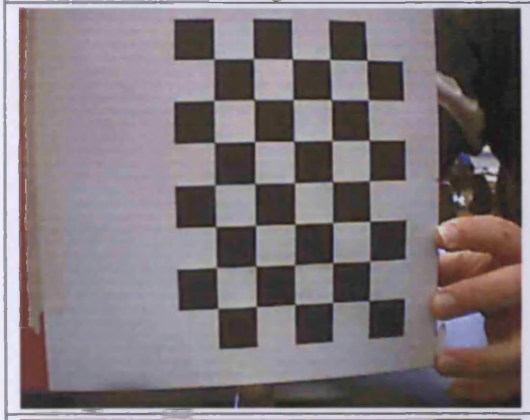
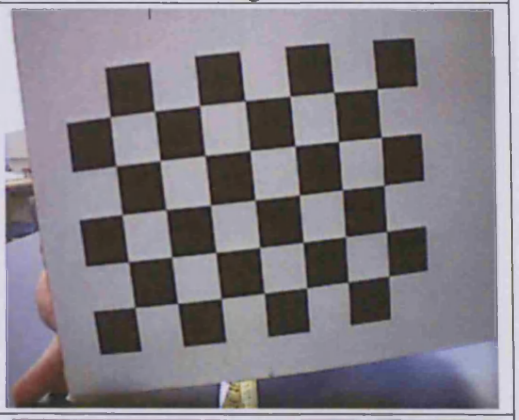
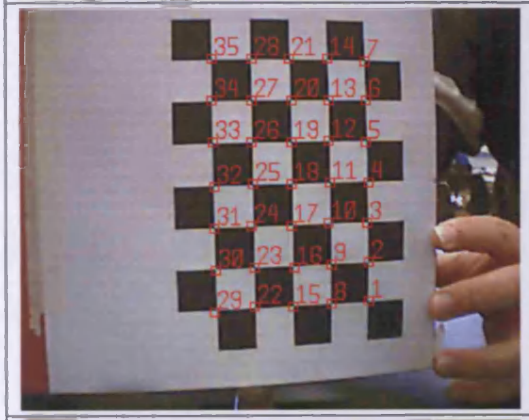
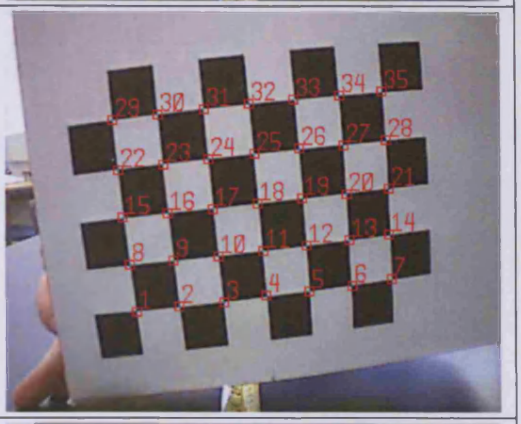
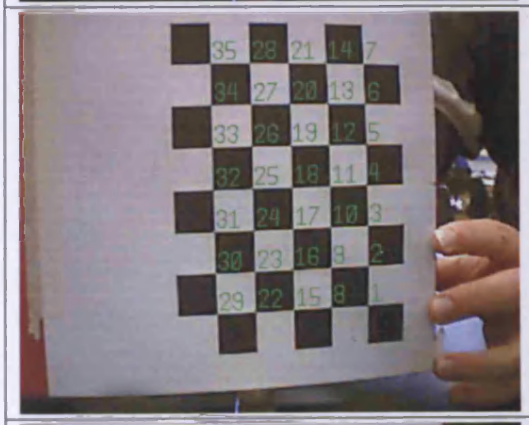
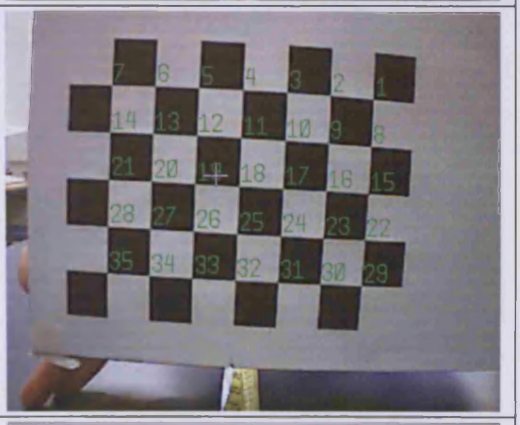
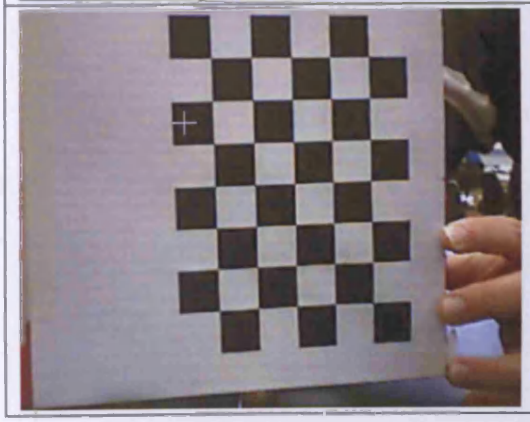
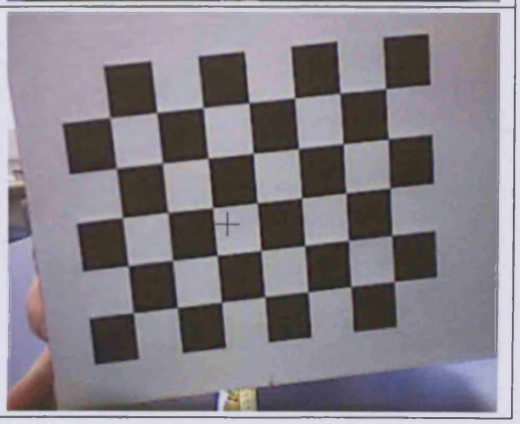
Image 7	Image 8
	
	
	
	

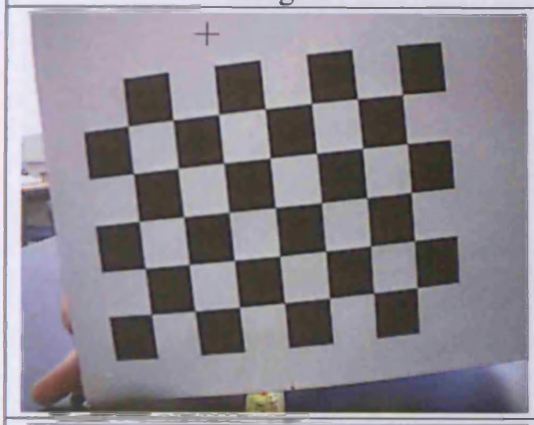
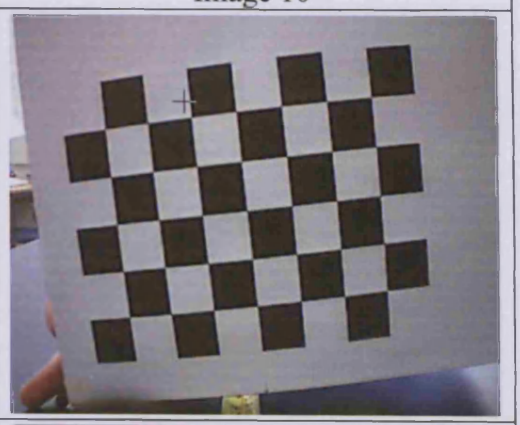
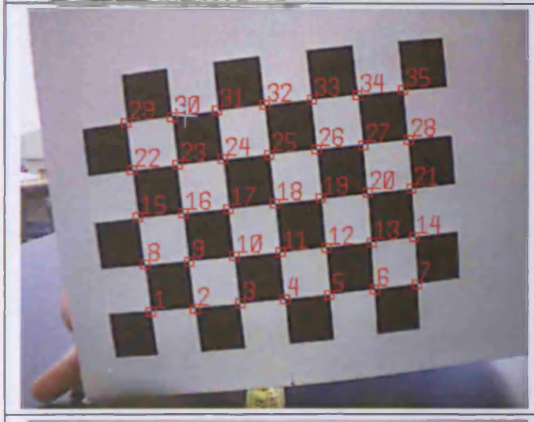
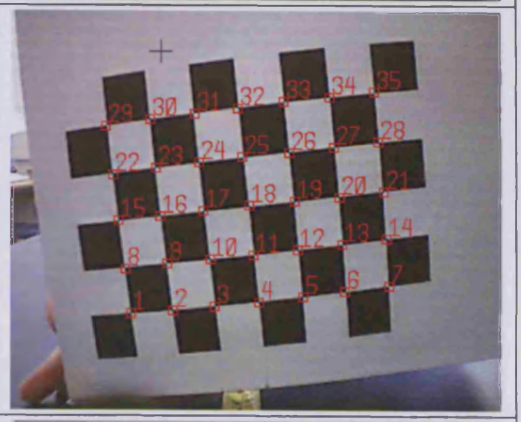
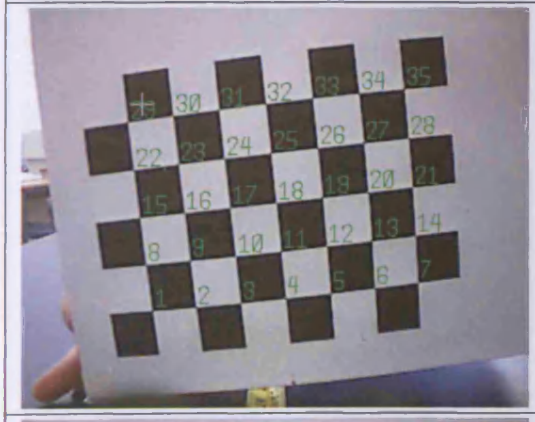
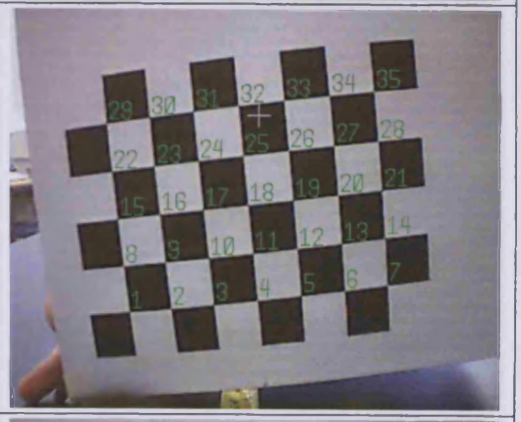
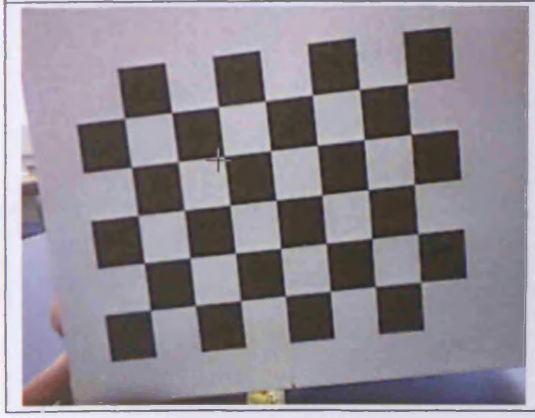
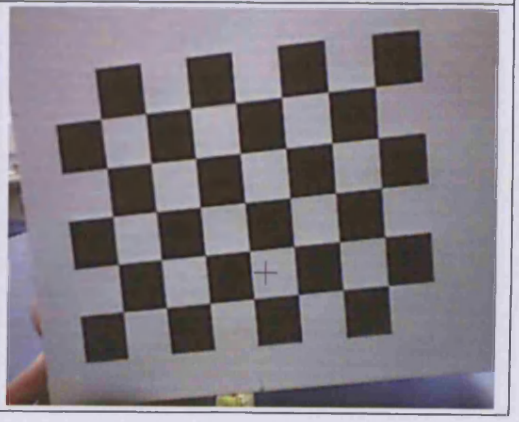
Image 9	Image 10
	
	
	
	

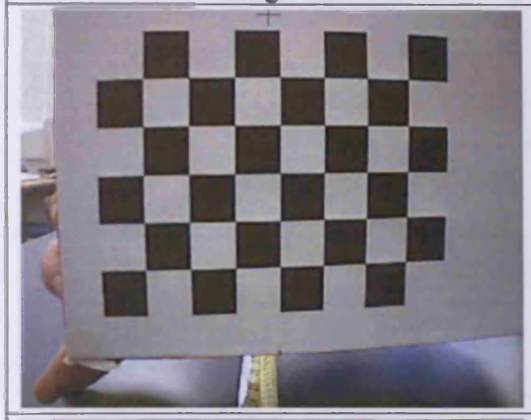
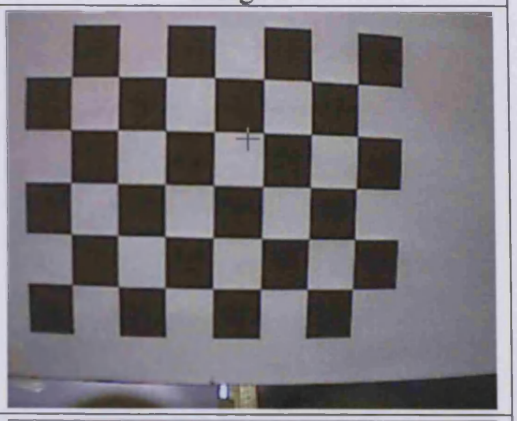
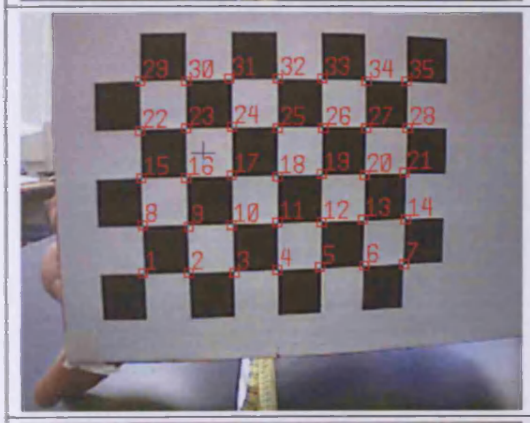
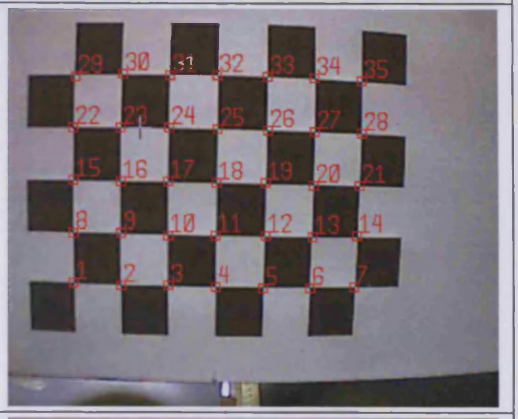
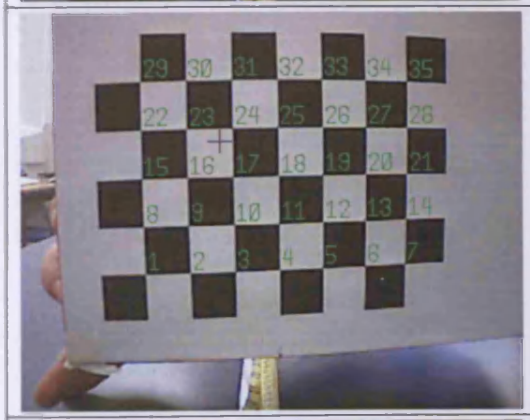
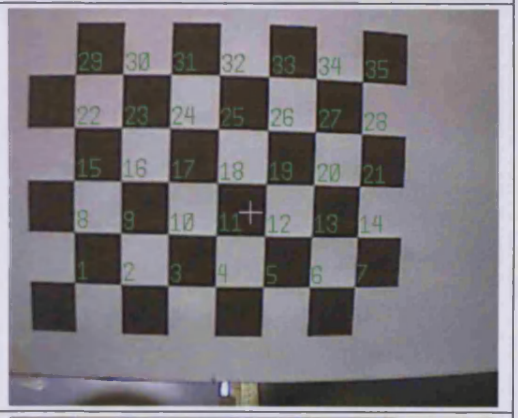
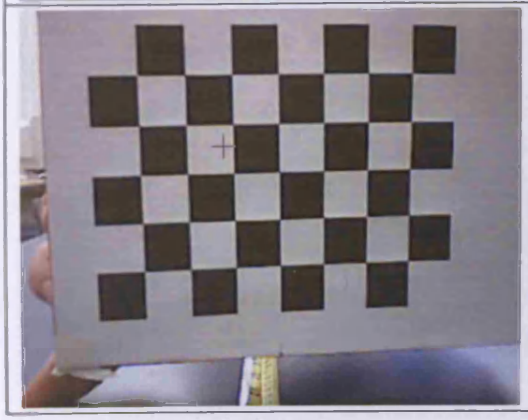
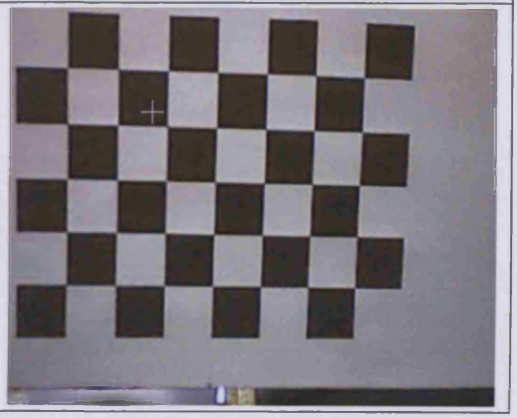
Image 11	Image 12
	
	
	
	

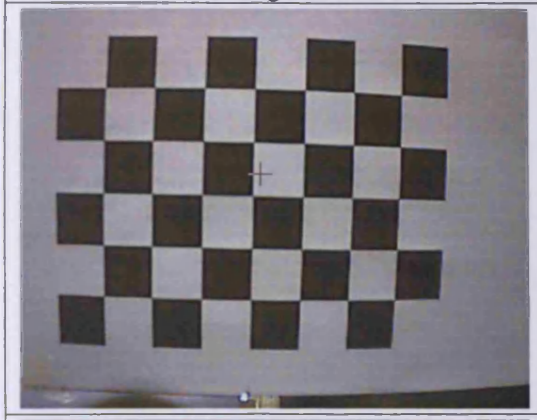
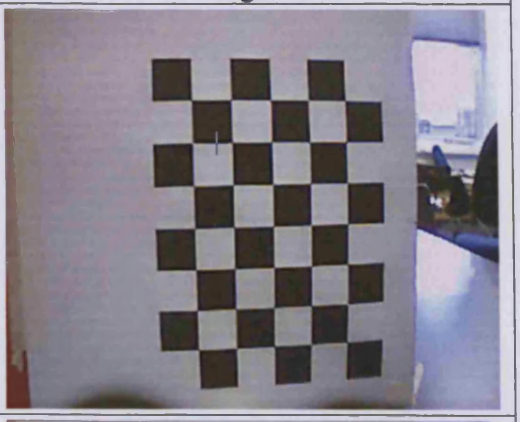
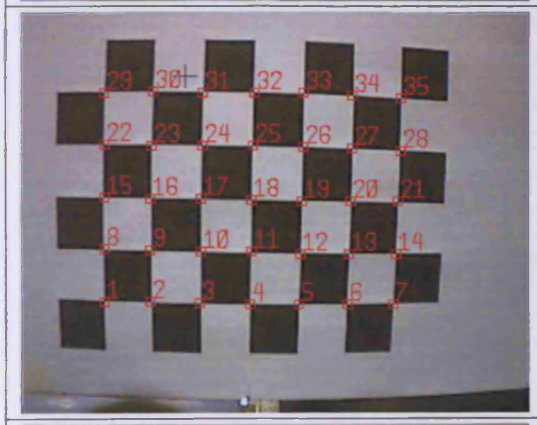
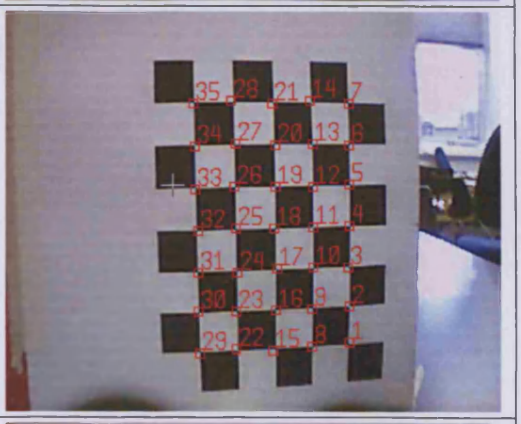
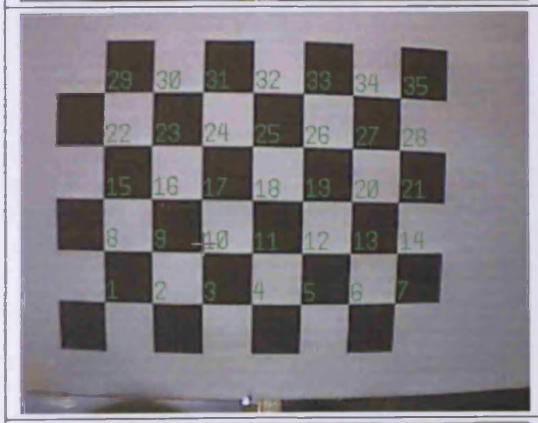
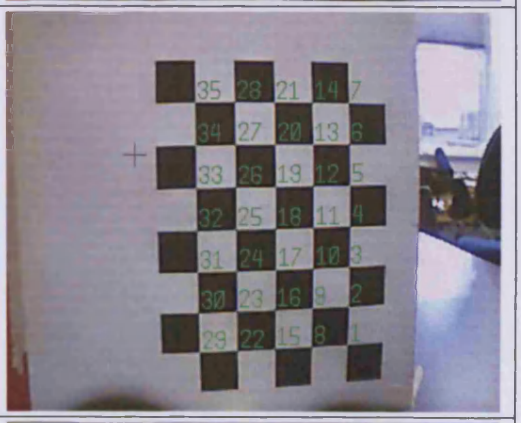
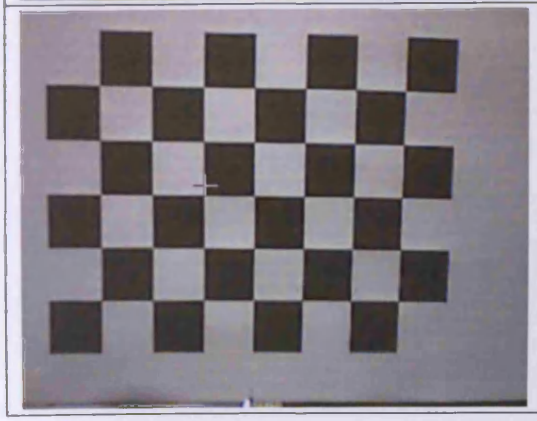
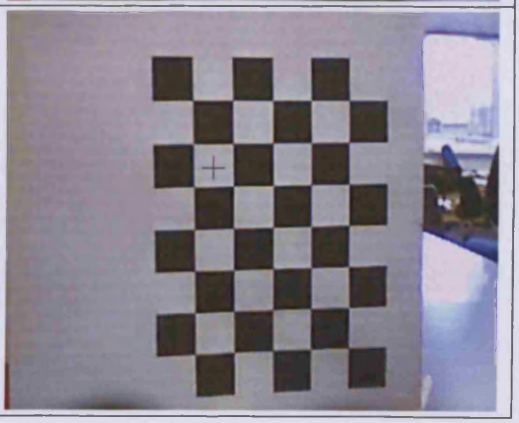
Image 13	Image 14
	
	
	
	

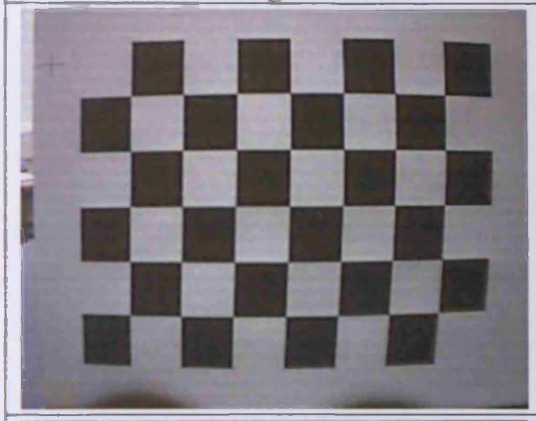
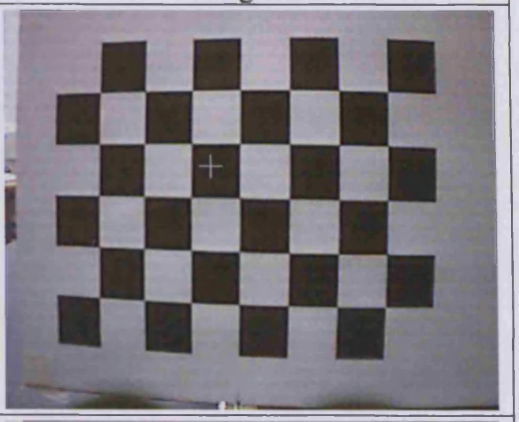
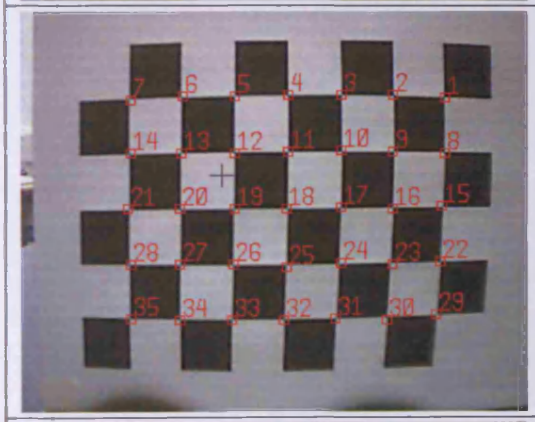
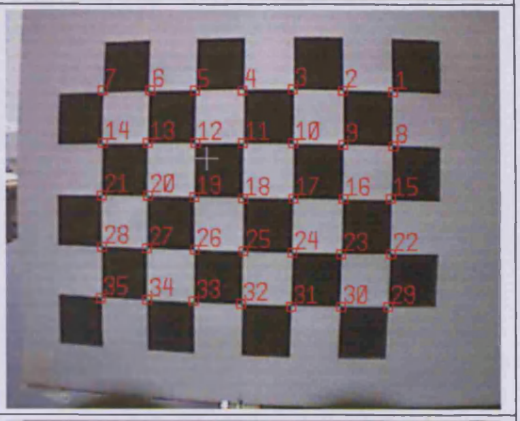
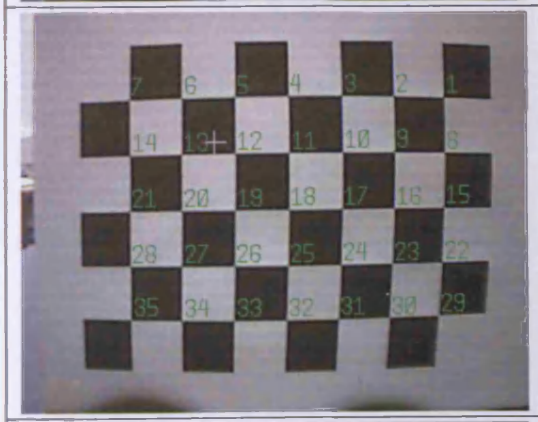
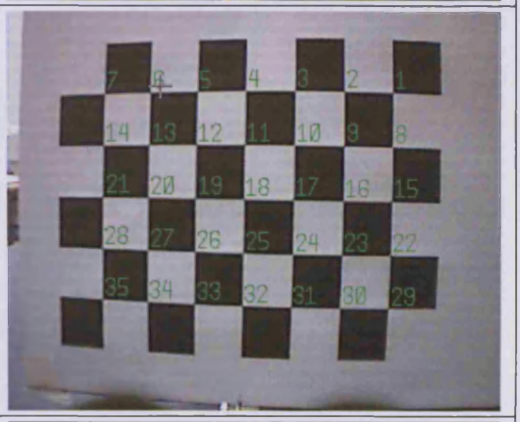
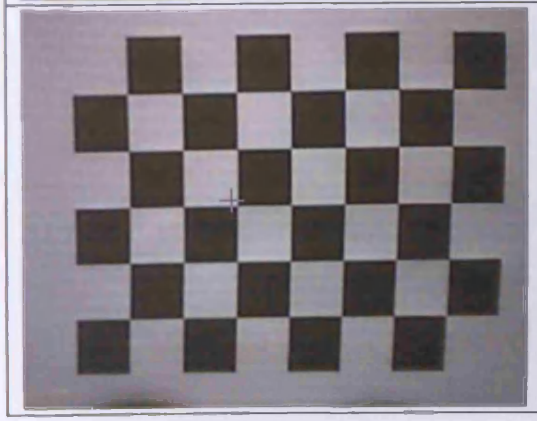
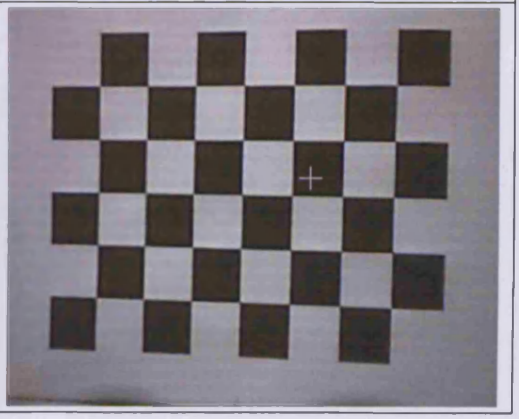
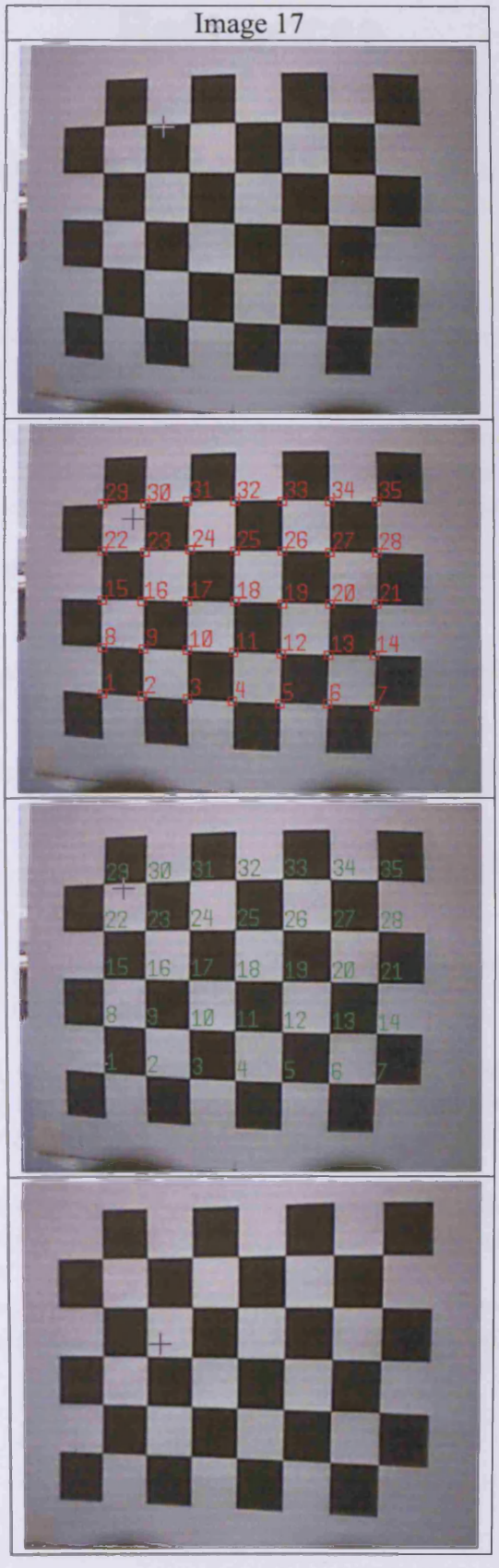
Image 15	Image 16
	
	
	
	

Image 17


29 30 31 32 33 34 35
22 23 24 25 26 27 28
15 16 17 18 19 20 21
8 9 10 11 12 13 14
1 2 3 4 5 6 7

References

Adams M. D., 2002. Coaxial range measurement - current trends for mobile robotic applications. *IEEE Sensors Journal.* Volume 2 (Issue 1), pp. 2-13.

al-Jazari I. a.-R., 1975. *The Book of Knowledge of Ingenious Mechanical Devices. (tranlated by Donald R. Hill).* Boston, United States of America: Reidel Publishing.

Almansa A., Desolneux A. and Vamech S., 2003. Vanishing point detection without any a priori information. *IEEE Transactions on Pattern Analysis and Machine Intelligence.* Volume 25 (Issue 4), pp. 502- 507.

Anderson M. L., 2003a. Embodied Cognition: A field guide. *Artificial Intelligence.* Volume 149 (Issue 1), pp. 91-130.

Anderson M. L., 2003b. Representations, symbols, and embodiment. *Artificial Intelligence.* Volume 143 (Issue 1), pp. 151-156.

Asimov I., 1942. *Runaround.* New York, United States of America: Faucett Crest.

Asimov I., 1950. *I, Robot.* Garden City, New York, United States of America: Doubleday.

Aslantas V., 1997. *New techniques for determining depth from defocusing.* PhD Thesis. Cardiff University.

Aslantas V., 2001. Criterion Functions for Automatic Focussing. *In: Xth Turkish Symptoium on Artificial Intelligence and Neural Networks (TAINN2001).* Gazimagusa, Turkey, 21-22 June 2001. pp. 301-311.

Aslantas V. and Tunçkanat M., 2003. Estimation of Depth from Defocusing Using a Neural Network. *In: International Journal of Computational Intelligence,*
Proceedings of the International Conference on Signal Processing (ICSP 2003). Çanakkale, Turky, September 24-26 2003. pp. 260-265.

Balkcom D. and Mason M., 2000. Time Optimal Trajectories for Bounded Velocity Differential Drive Robots. *In: IEEE International Conference on Robotics and Automation (ICRA '00)*. San Francisco, California, United States of America, 24-28 April 2000. pp. 2499-2504.

Barsalou L. W., Niedenthal P. M., Barbey A. and Ruppert J., 2003. Social embodiment. *The Psychology of Learning and Motivation.* Volume 43, pp. 43-92.

Beer R. D., 1995. A dynamical systems perspective on agent-environment interaction. *Artificial Intelligence.* Volume 72 (Issue 1-2), pp. 173-215.

Beutler J., 1998. *The BOM Project*. Technical Report. University of Wales, Cardiff.

Borenstein J., Everett H. R. and Feng L., 1996. *Navigating Mobile Robots: Systems and Techniques*. Wellesley, Mass.: A. K. Peters, Ltd.

Braitenberg V., 1984. *Vehicles: Experiments in Synthetic Psychology.* Cambridge, Mass.: MIT Press.

Brooks R. A., 1986. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation.* Volume 2 (Issue 1), pp. 14-23.

Brooks R. A., 1991a. Intelligence Without Reason. *In:* J. Myopoulos and R. Reiter Editors. *Proceedings of the 12th International Joint Conference on Artificial Intelligence (IJCAI-91)*. Darling Harbour, Sydney, Australia, 24-30 August 1991. pp. 569-595.

Brooks R. A., 1991b. Intelligence Without Representation. *Artificial Intelligence.* Volume 47 (Issue 1-3), pp. 139-159.

Brooks R. A., 2002. *Flesh and Machines: How Robots Will Change Us.* Pantheon Books.

Brown M. Z., Burschka D. and Hager G. D., 2003. Advances in computational stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence.* Volume 25 (Issue 8), pp. 993-1008.

Cantoni V., Lombardi L., Porta M. and Vallone U., 2001. Qualitative Estimation of Depth in Monocular Vision. *In: Proceedings of the 4th International Workshop on Visual Form (IWVF4)*. Capri, Italy, May 28-30 2001. Springer-Verlag, series "Lecture Notes in Computer Science". pp. 135-145.

Cao A. and Borenstein J., 2002. Experimental Characterization of Polaroid Ultrasonic Sensors in Single and Phased Array Configuration. *In: UGV Technology Conference at the 2002 SPIE AeroSense Symposium.* Orlando, Florida, United States of America, 1-5 April 2002.

Capek K., 1921. *R.U.R. (Rossum's Universal Robots).* (out of print).

Caprile B. and Torre V., 1990. Using vanishing points for camera calibration. *International Journal of Computer Vision.* Volume 4 (Issue 2), pp. 127-140.

Cavaliere R., 1996. *The Mechanical Construction of a Small Mobile Robot.* Technical Report. University of Wales, Cardiff.

Chrisley R., 2003. Embodied Artificial Intelligence. *Artificial Intelligence.* Volume 149 (Issue 1), pp. 131-150.

Clancey W. J., 1997. *Situated Cognition: On Human Knowledge and Computer Representations.* Cambridge University Press:

Clark A., 1997. *Being there: putting brain, body, and world together again.* Cambridge, Mass.: MIT Press.

Clarke T. A. and Fryer J. G., 1998. The Development of Camera Calibration Methods and Models. *The Photogrammetric Record.* Volume 16 (Issue 91), pp. 51-66.

Consortium P. E., 2001. *PC/104 Specification.* PC/104 Embedded Consortium. (Version 2.4).

Corre J., 2001. *The Improved BOM Robots.* Technical Report. University of Wales, Cardiff.

Criminisi A., 1999. *Accurate Visual Metrology from Single and Multiple Uncalibrated Images.* Ph.D. Thesis. University of Oxford.

Criminisi A., 2001. *Accurate Visual Metrology from Single and Multiple Uncalibrated Images.* London: Springer-Verlag Ltd.

Criminisi A., Reid I. and Zisserman A., 2001. Single View Metrology. *International Journal of Computer Vision.* Volume 40 (Issue 2, November), pp. 123-148.

Crisman J. D. and Webb J. A., 1991. The Warp machine on Navlab. *IEEE Transactions on Pattern Analysis and Machine Intelligence.* Volume 13 (Issue 5), pp. 451-465.

Dautenhahn K., 1997. Biologically Inspired Robotic Experiments on Interaction and Dynamic Agent-Environment Couplings. *In: Selbstorganisation von Adaptivem Verhalten (SOAVE'97).* Technische Universität Ilmenau, Germany, 23-24 September 1997. Fortschrittberichte VDI, Reihe 8, Nr. 663. pp. 14-24.

Dautenhahn K., 1999. Embodiment and Interaction in Socially Intelligent Life-Like Agents. *In: C. L. Nehaniv Editor. Computation for Metaphors, Analogy and Agents.* Springer. pp. 102-142. Springer Lecture Notes in Artificial Intelligence, Volume 1562.

Dautenhahn K., Ogden B. and Quick T., 2002. From embodied to socially embedded agents - implications for interaction-aware robots. *Cognitive Systems Research. Special issue on Situated and Embodied Cognition, Ziemke, T. Guest-Editor.* Volume 3 (Issue 3), pp. 397-428.

Descartes R., 1641. *Meditationes de prima philosophia, in quibus dei existentia, & animæ humanæ à corpore distinctio, demonstrantur.* Parisiis: Michaelem Soly.

Descartes R., 1664. *De homine.* Amstelodami: Danielem Elseverium.

Deschênes F., Ziou D. and Fuchs P., 2004. An unified approach for a simultaneous and cooperative estimation of defocus blur and spatial shifts. *Image Vision Computing.* Volume 22 (Issue 1), pp. 35-57.

Devantech, 2004. *Devantech Ltd. website.* Available from: http://www.robot-electronics.co.uk/ [Last accessed on the 14 March 2004].

Djakov V., 1997. *An Intelligent Sensor for Direction Detection.* M.Sc. Thesis. University of Wales, Cardiff.

Dreyfus H. L., 1991. *Being-in-the-World: A Commentary on Heidegger's Being and Time, Division I.* Cambridge, Mass.: MIT Press.

Duffy B. R. and Joue G., 2000. Intelligent Robots: The Question of Embodiment. *In: BRAIN-MACHINE'2000.* Ankara, Turkey, 20-22 December 2000.

Encyclopædia Britannica, 2004a. *Archytas of Tarentum.* Available from: http://www.britannica.com/eb/article?eu=9411 [Last accessed on the 21st June 2004].

Encyclopædia Britannica, 2004b. *Automaton.* Available from: http://www.britannica.com/eb/article?eu=11506 [Last accessed on the 21st June 2004].

Encyclopædia Britannica, 2004c. *Jacques de Vaucanson.* Available from: http://www.britannica.com/eb/article?tocId=9074908 [Last accessed on the 21st June 2004]

Encyclopædia Britannica, 2004d. *Manchester Mark I.* Available from: http://www.britannica.com/eb/article?tocId=216048 [Last accessed on the 21st June 2004]

Etzioni O., 1993. Intelligence without robots: a reply to Brooks. *AI Magazine.* Volume 14 (Winter 1993) (Issue 4), pp. 7-13.

Faugeras O. D., 1992. What can be seen in three dimensions with an uncalibrated stereo rig? *In: Proceedings of the Second European Conference on Computer Vision (ECCV '92).* Santa Margherita Ligure, Italy, 19-22 May 1992. pp. 563-578.

Faugeras O. D., 1993. *Three-Dimensional Computer Vision.* Cambridge, Mass.: MIT Press.

Fikes R. E., Hart P. E. and Nilsson N. J., 1972. Learning and Executing Generalized Robot Plans. *Artificial Intelligence.* Volume 3 (Issue 4), pp. 251-288.

Franklin S., 1997. Autonomous agents as embodied AI. *Cybernetics & Systems.* Volume 28 (Issue 6), pp. 499-520.

Franklin S. and Graesser A., 2001. Modeling Cognition with Software Agents. *In: Proceedings of the 23rd Annual Conference of the Cognitive Science Society (CogSci2001).* Edinburgh, Scotland, United Kingdoms, 1-4 August 2001. pp. 301-306.

Fujita M. and Kageyama K., 1997. An Open Architecture for Robot Entertainment. *In: Proceedings of the First International Conference on Autonomous Agents.* Marina del Rey, California, United States of America, 05-08 February 1997. pp. 435-442.

Fujita M., Kitano H. and Kageyama K., 1997. A Legged Robot for RoboCup Based on "OPENR". *In: the First International Workshop on RoboCup in conjunction with RoboCup-97 at IJCAI-97.* Nagoya Congress Center, Nagoya, Japan, 23 August 1997. pp. 168-180.

Gaukroger S., 1995. *Descartes: an intellectual biography.* Oxford: Clarendon Press, New York: Oxford University Press.

Gerkey B. P. and Mataric M. J., 2002. Sold!: Auction Methods for Multi-Robot Coordination. *IEEE Transactions on Robotics and Automation, Special Issue on Advances in Multi-Robot Systems.* Volume 18 (Issue 5), pp. 758-786.

Grossman P., 1987. Depth from focus. *Pattern Recognition Letters.* Volume 5 (Issue 1), pp. 63-70.

Hallam J. C. and Malcolm C. A., 1994. Behavior - Perception, Action and Intelligence - The View from Situated Robotics. *Philosophical Transactions of the Royal Society: Physical Sciences and Engineering.* Volume A 349 (Issue 1689), pp. 29-42.

Haugeland J., 1985. *Artificial intelligence : the very idea.* Cambridge Mass. And London: MIT Press.

Hebert M., 2000. Active and passive range sensing for robotics. *In: IEEE International Conference on Robotics and Automation (ICRA '00).* San Francisco, California, United States of America, 24-28 April 2000. pp. 102 - 110.

Heikkila J. and Silven O., 1997. A Four-step Camera Calibration Procedure with Implicit Image Correction. *In: Conference on Computer Vision and Pattern Recognition (CVPR '97).* San Juan, Puerto Rico., June 17-19 1997. IEEE Computer Society, Washington, DC, USA.

Hendriks-Jansen H., 1996. *Catching Ourselves in the Act: Situated Activity, Interactive Emergence, Evolution, and Human Thought.* Cambridge, Mass.: MIT Press.

Holland O., 2003. Exploration and high adventure: the legacy of Grey Walter. *Philosophical Transactions of the Royal Society: Mathematical, Physical and Engineering Sciences.* Volume 361 (Issue 1811), pp. 2085 - 2121.

Holland O. E., 1997. Grey Walter: The Pioneer of Real Artificial Life. *In:* C. Langton Editor. *Proceedings of the 5th International Workshop on Artificial Life*. Nara, Japan, 16-18 May 1996. MIT Press, Cambridge. pp. 34-44.

Homer, 2002. *The Iliad, a new translation by Ian Johnston.*

Honda, 2004. *ASIMO Web site*. Available from: http://world.honda.com/ASIMO/ [Last accessed on the 17th June 2004].

Husbands P., Harvey I. and Cliff D., 1993. An Evolutionary Approach to Situated AI. *In:* A. Sloman, D. Hogg, G. Humphreys, A. Ramsay and D. Partridge Editors. *Prospects for Artificial Intelligence: Proceedings of the Artificial Intelligence and Cognitive Science Conference (AISB93), the Ninth Biennial Conference of the Society for the Study of Artificial Intelligence and the Simulation of Behaviour*. The University of Birmingham, 29 March-2 April 1993. Amsterdam: IOS Press. pp. 61-70.

Hyde T., 1700. *Historia religionis veterum persarum*. Editio Nova. Oxford: E. Theatro Sheldoniano.

IEEE, 1983. *IEEE Standard Microcomputer System Bus*. IEEE Standard Association. (0-7381-2753-1).

Kanade T., Okutomi M. and Nakahara T., 1992. A Multiple-baseline Stereo Method. *In: Proceedings of the DARPA Image Understanding Workshop*. San Diego, California, United States of America, 26-29 January 1992. pp. 409-426.

Kant I., 1798. *Anthropologie in pragmatischer Hinsicht*. 1996 Edition. Southern Illinois: University Press.

Keijzer F., 2002. Representation in dynamical and embodied cognition. *Cognitive Systems Research. Special issue on Situated and Embodied Cognition, Ziemke, T. Guest-Editor.* Volume 3 (Issue 3), pp. 275-288.

Kellis E., 2002. *An Evaluation of the Scientific Potential of Evolutionary Artificial Life God-Games: Considering an Example Model for Experiments and Justification*. M.Sc. Thesis. University of Sussex.

Kim T., Seo Y. and Hong K.-S., 1998. Physics-based 3D Position Analysis of a Soccer Ball from Monocular Image Sequences. *In: In Proceeding of the 6th International Conference on Computer Vision (ICCV 1998).* Bombay, India, pp. 721-726.

Kimura S., Shinbo T., Yamaguchi H., Kawamura E. and Naka K., 1999. A Convolver-Based Real-Time Stereo Machine (SAZAN). *In: Conference on Computer Vision and Pattern Recognition (CVPR '99).* Fort Collins, Colorado, United Sates of America, 23-25 June 1999. pp. 457-463.

Konolige K., 1997. Small Vision Systems: Hardware and Implementation. *In: Eighth International Symposium on Robotics Research (ISRR '97).* Hayama, Japan, October 1997. Springer-Verlag. p. 203–212.

Konolige K., Ortiz C., Vincent R. and Andrew A., 2002. *DARPA Software for Distributed Robotics.* Technical Report. SRI International.

Krotkov E., 1987. Focusing. *International Journal of Computer Vision.* Volume 1 (Issue 3), pp. 223-237.

Kushmerick N., 1997. Software agents and their bodies. *Minds & Machines.* Volume 7, pp. 227-247.

Lakoff G. and Johnson M., 1999. *Philosophy in the Flesh: The Embodied Mind and Its Challenge to Western Thought.* Basic Books.

Lambert J. M., Morre B. and Ahmadi M., 2001. Essential Real-Time and Modeling Tools for Robot Rapid Prototyping. *In: 6th International Symposium on Artificial Intelligence, Robotics, and Automation in Space (iSAIRAS).* Montreal, Canada, 18-21 June 2001.

Lenser S. and Veloso M., 2003. Visual sonar: Fast obstacle avoidance using monocular vision. *In: IEEE/ISJ International Conference on Intelligent Robots and Systems (IROS'03).* Bally's Las Vegas Hotel, Nevada, United States of America, , October 27 - 31, 2003.

Levesque H. and Pagnucco M., 2000. Legolog: Inexpensive Experiments in Cognitive Robotics. *In: Cognitive Robotics Workshop at ECAI.* Berlin, Germany, 21-22 August 2000. pp. 104-109.

Lewis R. A. and Johnston A. R., 1977. A scanning laser rangefinder for a robotic vehicle. *In:* R. Reddy Editor. *5th International Joint Conference on Artificial Intelligence (IJCAI 1977)*. Cambridge, Massachuset, United States of America, William Kaufmann. pp. 762-768.

Li Y. F. and Chen S. Y., 2003. Automatic recalibration of an active structured light vision system. *IEEE Transactions on Robotics and Automation.* Volume 19 (Issue 2), pp. 259-268.

Lichtensteiger L. and Ralf S., 2000. The Evolution of an Artificial Compound Eye by Using Adaptive Hardware. *In: Proceedings of the 2000 Congress on Evolutionary Computation (CEC00)*. La Jolla Marriott Hotel, San Diego, California, United States of America, 16-19 July 2000. IEEE Press. pp. 1144-1151.

Liebowitz D. and Zisserman A., 1998. Metric rectification for perspective images of planes. *In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR'98)*. Santa Barbara, California, United States of America, June 23–25 1998. pp. 482-488.

Liedtke M., 1997. *Technical Documentation Distance Board.* Technical Report. University of Wales, Cardiff.

Locke J., 1690. *An essay concerning humane understanding.* London: T. Basset.

LoPresti E. F., Simpson R. C., Miller D. and Nourbakhsh I., 2002. Evaluation of Sensors for a Smart Wheelchair. *In: 25th Annual Conference on Rehabilitation Engineering (RESNA 2002).* Minneapolis, Minnesota, United States of America, June 2002. pp. 166-168.

Low K. H., Leow W. K. and Ang M. H. J., 2002. A hybrid mobile robot architecture with integrated planning and control. *In: Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 1*. Bologna, Italy, ACM Press. pp. 219-226.

Mackworth A. K., 1992. On Seeing Robots. *In: A. Basu and X. Li Editors. Computer Vision: Systems, Theory, and Applications*. Singapore: World Scientific Press. pp. 1-13.

Martin F. G., 1996. Kids Learning Engineering Science Using LEGO and the Programmable Brick. *In: Meeting of the American Educational Research Association*. New York, United States of America, 8-12 April 1996.

Mataric M. J., 2003. Situated Robotics. *In: L. Nadel Editor. Encyclopedia of Cognitive Science*. London: Nature Publishing Group, Macmillan Reference Limited.

Maturana H. R. and Varela F. J., 1980. *Autopoiesis and Cognition: The Realization of the Living*. Boston: D. Reidel Publishing Co.

MEC, 2002. *MEC Newsletter Autumn 2002*. Available from: http://www.mec.cf.ac.uk/services/docs/MEC%20NL%20autumn%202002.pdf [Last accessed on the 10th September 2003].

Mettenleiter M., Härtl F. and Fröhlich C., 2000. Imaging Laser Radar for 3-D Modelling of Real World Environments. *In: International Conference on OPTO / IRS2 / MTT,*. Erfurt, Germany, 9-11 November 200.

Michel O., 2003. *Webot User Guide*. Technical Report. (Version 3.2.21). Cyberbotics Ltd.

Miranda C. A., 2004. *The Sawrm Keeper, The Time Magazine Online Edition.* Available from: http://www.time.com/time/2004/innovators/200406/mclurkin.html [Last accessed on the 18 January].

Mondada F., Franzi E. and Ienne P., 1993. Mobile Robot Miniaturisation: A Tool for Investigation in Control Algorithms. *In: Proceedings of the Third International Symposium on Experimental Robotics*. Kyoto, Japan, 28-30 Octobre 1993. pp. 501-513.

Moravec H. P., 1980. *Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover*. Ph.D Thesis. Stanford University, Stanford, California, United Sates of America.

Moravec H. P., 1983. *The Stanford Cart and The CMU Rover*. Technical Report. The Robotics Institute, Carnegie-Mellon University.

Moravec H. P., 1998. *Robot, mere machine to transcendent mind*. Oxford: University Press.

Morris R., 1997. Deep Blue Versus Kasparov: The Significance for Artificial Intelligence, Collected Papers from the 1997 AAAI Workshop. *In: The 1997 AAAI Workshop*. AAAI Press. p. 68.

Müller J., 1834-40. *Handbuch der physiologie des menschen für vorlesungen.* Coblenz: J. Hölscher.

Nayar S. K. and Nakagawa Y., 1990. Shape from Focus: An Effective Approach for Rough Surfaces. *In: IEEE International Conference on Robotics and Automation (ICRA '90)*. Cincinnati, Ohio, United Sates of America, May 13-18, 1990. pp. 218-225.

New Scientist, 1998. Walter's World. *New Scientist*. 25 Jully. pp. 54-55.

Nilsson N. J., 1984. *Shakey the robot*. Technical Report. (Technical Report 323). Artificial Intelligence Center, SRI International, 333 Ravenswood Avenue, Menlo Park, CA 94025.

Nourbakhsh I., Andre D., Tomasi C. and Genesereth M., 1997. Mobile Robot Obstacle Avoidance via Depth from Focus. *Robotics and Autonomous Systems*. Volume 22, pp. 151-158.

O'Hare G. M. P. and Duffy B. R., 2002. Agent Chameleons: Migration and Mutation Within and Between Real and Virtual Spaces. *In: The Society for the Study of Artificial Intelligence and the Simulation of Behaviour (AISB 2002)*. Imperial College, England, 3-5 April 2002.

Oka N., Morikawa K., Komatsu T., Suzuki K., Hiraki K., Ueda K. and Omori T., 2001. Embodiment Without a Physical Body. *In: The Proceedings of the International Workshop of Developmental Embodied Cognition (DECO2001) in the 23rd Annual Meeting of the Cognitive Science Society (CogSci2001)*. Edinburgh, Scotland, United Kingdoms, 1-4 August 2001. pp. 48-52.

Palacin J., Lasa X. and Marco S., 2003. Straight-Line Path Following in Cleaning Robots Using Lateral Ultrasonic Sensors. *In: IEEE Instrumentation and Measurement Technology Conference (IMTC)*. Vail, Colorado, United States of America, 20-22 May 2003. pp. 1484-1487.

Peer P. and Solina F., 2002. Panoramic Depth Imaging: Single Standard Camera Approach. *International Journal of Computer Vision*. Volume 47 (Issue 1-3), pp. 49-160.

Pfeifer R. and Scheier C., 2001. *Understanding intelligence*. 2nd printing (paperback edition). Cambridge, Mass.: MIT Press.

Pham D. T. and Parhi D. R., 2003. Navigation of Multiple Mobile Robots Using a Neural Network and a Petri Net Model. *Robotica.* Volume 21 (Issue 1), pp. 79-93.

Piaget J., 1946. *Naissance de l'intelligence chez l'enfant. (The origins of intelligence in children, translated by Margaret Cook in 1952)*. New York: International Universities Press.

Plato, 360 BC. *The republic*. 2003 Edition. Penguin Books.

Prinz W., 1987. Ideomotor action. *In: H. Heuer and A. F. Sanders Editors. Perspectives on perception and action*. Hillsdale, NJ: Erlbaum. pp. 47-76.

Quick T. and Dautenhahn K., 1999. Making Embodiment Measurable. *In: 4th Fachtagung der Gesellschaft für Kognitionswissenschaft; Workshop on Embodied Mind and ALife (KogWis'99)*. Bielefeld, Germany, 28 September - 1 October 1999.

Quick T., Dautenhahn K., Nehaniv C. L. and Roberts G., 1999a. The Essence of Embodiment: A Framework for Understanding and Exploiting Structural Coupling Between System and Environment. *In: Proceedings of the Third International Conference on Computing Anticipatory Systems (CASYS'99)*. Liège, Belgium, 9-14 August 1999.

Quick T., Dautenhahn K., Nehaniv C. L. and Roberts G., 1999b. On Bots and Bacteria: Ontology Independent Embodiment. *In: European Conference on Artificial Life*. Swiss Federal Institute of Technology, Lausanne, Switzerland, 13-17 September 1999. pp. 339-343.

Reshko G., Mason M. and Nourbakhsh I., 2002. *Rapid Prototyping of Small Robots*. Technical Report. (CMU-RI-TR-02-11). Robotics Institute, Carnegie Mellon University, Pittsburgh.

Resnick M., Martin F., Sargent R. and Silverman B., 1996. Programmable Bricks: Toys to Think With. *IBM Systems Journal.* Volume 35 (Issue 3-4), pp. 443-452.

Risku H., 2002. Situatedness in translation studies. *Cognitive Systems Research.* Volume 3 (Issue 3), p. 523–533.

Robbins S. E., 2002. Semantics, experience and time. *Cognitive Systems Research. Special issue on Situated and Embodied Cognition, Ziemke, T. Guest-Editor.* Volume 3 (Issue 3), pp. 301-337.

Rus D., Kabir A., Kotay K. and Soutter M., 1996. Guiding Distributed Manipulation with Mobile Sensors. *In: Proceedings of the International Conference on Multi Sensor Fusion and Integration.* Washington DC, United States of America, 8-11 December 1996.

Ryle G., 1949. *The concept of mind.* London: Hutchinson & Company.

Sahota M. K. and Mackworth A. K., 1994. Can Situated Robots Play Soccer? *In: Proceedings of Canadian Artificial Intelligence Conference (Canadian AI-94).* Banff, Alberta, Canada, 16-20 May 1994. pp. 249-254.

Schomaker H., 1996. *The Relative Robot Direction Device (R2D2).* Technical Report. University of Wales, Cardiff.

Se S. and Brady M., 2003. Road feature detection and estimation. *Machine Vision and Applications.* Volume 14 (Issue 3), pp. 157-165.

Sharkey N. E. and Ziemke T., 2001. Mechanistic versus phenomenal embodiment: Can robot embodiment lead to strong AI? *Cognitive Systems Research.* Volume 2 (Issue 4), pp. 251-262.

Simon H. A., 1957. *Administrative behavior: a study of decision-making processes in administrative organization.* Second Edition. New York: Free Press, London: Collier-Macmillan.

Siuru B., 1994. The smart vehicles are here. *Popular Electronics.* 1. 1. pp. 41-45.

ST.Microelectronics, 1998. *VV6404 Coulour CMOS sensor Datasheet.* Available from: http://www.mpja.com/download/12513st.pdf [Last accessed on the 7th June 2004].

Suchman L. A., 1987. *Plans and Situated Actions: The problem of human machine communication.* Cambridge: Cambridge University Press.

Terada K., Takayuki N., Hideaki T. and Tsukasa O., 2001. Embodiment Based Object Recognition for Vision-Based Mobile Agents. *Journal of Robotics and Mechatoronics.* Volume 13 (Issue 1), p. 88–95.

Tovar B., Murrieta-Cid R. and Esteves C., 2002. Robot motion planning for map building. *In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2002).* Lausanne, Switzerland, September 30 - October 4.

Tsai R. Y., 1987. A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses. *IEEE Journal of Robotics and Automation.* Volume 3 (Issue 4), pp. 323-344.

Varela F. J., Thompson E. and Rosch E., 1991. *The Embodied Mind: Cognitive Science and Human Experience.* Cambridge, Mass: MIT Press.

Vision, 1998. *VISION CPiA Data Sheet (VV0670P001).* Available from: http://webcam.sourceforge.net/docs/cpia_datasheet4.pdf [Last accessed on the 07th June 2004].

von Wolff C. F., 1734. *Psychologia rationalis.* New York: Olms.

Walter W. G., 1950. An Imitation of Life. *Scientific American.* Volume 182 (Issue 5, May), pp. 42-45.

Walter W. G., 1951. A Machine that Learns. *Scientific American.* Volume 185 (Issue 8, August), pp. 60-63.

Walter W. G., 1963. *The Living Brain.* New York: W. W. Norton.

Warwick K., Kelly I., Goodhew I. and Keating D. A., 1995. Behaviour and Learning in Completely Autonomous Mobile Robots. *In: Design and Development of Autonomous Agents, IEE Colloquium.* 21 November 1995. pp. 7/1-7/4.

Watson R. A., Ficici S. G. and Pollack J. B., 1999. Embodied Evolution: Embodying an Evolutionary Algorithm in a Population of Robots. *In:* P. J. Angeline, Z. Michalewicz, M. Schoenauer, X. Yao and A. Zalzala Editors. *Proceedings of the Congress on Evolutionary Computation.* Mayflower Hotel, Washington D.C., United States of America, 6-9 July 1999. IEEE Press. pp. 335-342.

Webb J. A., 1993. Implementation and performance of fast parallel multi-baseline stereo vision. *In: Proceedings of the DARPA Image Understanding Workshop.* Washington, D.C., United States of America, April 1993. pp. 1005-1012.

Werger B. B. and Mataric M. J., 1999. *Exploiting Embodiment in Multi-Robot Teams, IRIS Technical Report.* Technical Report. (IRIS-99-378). Institute for Robotics and Intelligent Systems, School of Engineering, University of Southern California.

Wetteborn H., 1993. *Laserabstandser-mittlungsvorrichtung.*

Wilberg J. and Siegberg A., 1998. *Experimental Robotics.* (Unpublished).

Wildes R. P., 1991. Direct recovery of three-dimensional scene geometry from binocular stereo disparity. *IEEE Transactions on Pattern Analysis and Machine Intelligence.* Volume 13 (Issue 8), pp. 761-774.

Wilson M., 2002. Six Views of Embodied Cognition. *Psychological Bulletin & Review.* Volume 9 (Issue 4, December), pp. 625-636.

Zhang Z., 1999. Flexible Camera Calibration By Viewing a Plane From Unknown Orientations. *In: International Conference on Computer Vision (ICCV'99).* Corfu, Greece, Septembre 1999. pp. 666-673.

Zhang Z., 2000. A Flexible new Technique for Camera Calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence.* Volume 22 (Issue 11), pp. 1330-1334.

Ziemke T., 1999. Rethinking Grounding. *In: A. Riegler, M. Peschl and A. von Stein Editors. Understanding Representation in the Cognitive Sciences - Does Representation Need Reality?* New York: Plenum Press, Kluwer Academic Publishers. pp. 177-190.

Ziemke T., 2001a. Are Robots Embodied? *In:* C. Balkenius, C. Breazeal, K. Dautenhahn, H. Kozima and J. Zlatev Editors. *Proceedings of the First International Workshop on Epigenetic Robotics: Modeling Cognitive Development in Robotic Systems.* Lund University Cognitive Science, Sweden, 17-18 September 2001. pp. 75-93.

Ziemke T., 2001b. Disentangling Notions of Embodiment. *In:* L. W. Pfeifer Editor. *Proceedings of the Workshop on Developmental Embodied Cognition (DECO-2001) at the 23rd Annual Meeting of the Cognitive Science Society (CogSci 2001).* Edinburgh, Scotland, United Kingdoms, 1-4 August 2001.

Zitnick C. L. and Kanade T., 2000. A cooperative algorithm for stereo matching and occlusion detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence.* Volume 22 (Issue 7), pp. 675-684.

Zlatev J., 1997. *Situated embodiment: Studies in the emergence of spatial meaning*. Ph.D. Thesis. Stockholm University.

Zlatev J., 2001. The epigenesis of meaning in human beings, and possibly in robots. *Minds and Machines*. Volume 11 (Issue 2), pp. 155-195.

Zlatev J., 2003. Polysemous or generality? Mu. *In: H. Cuyckens, R. Dirven and J. R. Taylor Editors. Cognitive Approaches to Lexical Semantics*. Berlin, Germany: Mouton de Gruyter. pp. 447-494.

Zou Y., Khing H. Y., Seng C. C. and Zhou C. C., 2000. Multi-ultrasonic Sensor Fusion for Mobile Robots. *In: IEEE Intelligent Vehicles Symposium (IV 2000)*. The Ritz-Carlton Hotel, Dearborn, Michigan, United States of America, October 4-5. pp. 387-391.